# Modellierung von Kontextinformationen für mobile Nutzungsszenarien Ein Ansatz zur Einbeziehung von sicherheitsrelevanter Benutzeraktivität

**MASTERTHESIS**

Masterstudiengang Medieninformatik


**FAKULTÄT FÜR INFORMATIK UND INGENIEURWISSENSCHAFTEN**

**TECHNISCHE HOCHSCHULE KÖLN**

Autor
Vitor Agostinho Guerreiro

Referent: Prof. Dr. Kristian Fischer
Korreferent: Prof. Dr. Stefan Karsch


Gummersbach, im Mai 2016

# Modeling Context Information in Mobile Usage Scenarios
# An Approach for Integration of security-relevant User Activity

**MASTER'S THESIS**

Master's program in Media Informatics


**FACULTY OF COMPUTER SCIENCE AND ENGINEERING SCIENCE**

**COLOGNE UNIVERSITY OF APPLIED SCIENCES**

Author
Vitor Agostinho Guerreiro

First Reviewer: Prof. Dr. Kristian Fischer
Second Reviewer: Prof. Dr. Stefan Karsch


Gummersbach, May 2016

Technology
Arts Sciences
TH Köln

Dipl.-Ing. (FH) Vitor Agostinho Guerreiro
Matrikel-Nr. / Register-No.: 11016455
E-Mail: vg@shee.org

Masterstudiengang: Medieninformatik
Web: https://www.medieninformatik.th-koeln.de/

Masterthesis zur Erlangung des akademischen Grades:
Master of Science (M.Sc.)
Eingereicht im Mai 2016.
Vorgelegt an der TH Köln (Technische Hochschule Köln).
Web: https://www.th-koeln.de/

Referent / First Reviewer:
Prof. Dr. rer. nat. Dipl.-Inform. Kristian Fischer
TH Köln - Institut für Informatik
Steinmüllerallee 1
51643 Gummersbach
E-Mail: kristian.fischer@th-koeln.de

Korreferent / Second Reviewer:
Prof. Dr. rer. nat. Dipl.-Inform. Stefan Karsch
TH Köln - Institut für Informatik
Steinmüllerallee 1
51643 Gummersbach
E-Mail: stefan.karsch@th-koeln.de

Das vollständige Manuskript zu dieser Masterthesis
wurde mittels Textsatzsystem TEX erstellt.

*Maria Antónia Guerreiro*

# Kurzbeschreibung


**Titel :** Modellierung von Kontextinformationen für mobile Nutzungsszenarien. Ein Ansatz zur Einbeziehung von sicherheitsrelevanter Benutzeraktivität.

**Autor :** Vitor Agostinho Guerreiro

**Referenten :** Prof. Dr. Kristian Fischer / Prof. Dr. Stefan Karsch

**Synopsis :** Um einen zuverlässigen kontext-sensitiven Sicherheitsdienst bereitzustellen, ist die Vollständigkeit des zur Bewertung genutzten Sicherheitskontextes von wesentlicher Bedeutung. Der Anwendungskontext leistet dazu einen fundamentalen Beitrag. Aufgrund der fehlenden Interpretationslogik ist allerdings der Anwendungskontext von *aussen* nicht ermittelbar. Die Integration einer den Sicherheitsdienst unterstützenden Komponente in die Anwendung schafft hierzu auf zweierlei Weise Abhilfe. Sie stellt Anwendungskontextinformationen zur Verfügung und gewährleistet effektiv eine kontext-sensitive Sicherheitsadaption.



**Stichwörter :** Sicherheitskontext, Kontextwahrnehmung, Kontextmodell, kontext-sensitive Sicherheit, Mobile Computing, Kontextmodellierung, Sicherheitsmassnahmen

**Datum :** Mai 2016

# Abstract

**Title :** Modeling Context Information in Mobile Usage Scenarios. An Approach for Integration of security-relevant User Activity.

**Author :** Vitor Agostinho Guerreiro

**Reviewers :** Prof. Dr. Kristian Fischer / Prof. Dr. Stefan Karsch

**Abstract :** The completeness of the security context is crucial to provide a reliable context-aware security service. The application context makes a fundamental contribution. Due to the lack of interpretation logic the application context can not be determined from outside. The integration of a security component into the application helps here in two ways. It provides application context information and ensures effective context-aware security adaptation.

**Keywords :** security context, context awareness, context model, context-aware security, mobile computing, context modeling, security measures

**Date :** May 2016

# Danksagung

An dieser Stelle möchte ich ganz besonders Prof. Dr. Kristian Fischer für die Anregung zu der untersuchten Thematik und dem großzügigen Gestaltungsspielraum bei der Bearbeitung danken. Herrn Prof. Dr. Stefan Karsch gilt mein besonderer Dank für sein Interesse an meiner Arbeit und für die freundliche Übernahme des Korreferats.

# Inhaltsverzeichnis

# Abbildungsverzeichnis

# Quelltextverzeichnis

# Abkürzungs- und Wortverzeichnis

| | |
|---|---|
| $t_n$ | Index, Zeitpunkt $n$ |
| *AssessSeccon* | Bewertung des Sicherheitskontextes via *SeclevelClass* |
| *Capabilities* | Konfiguration der Anwendung bezüglich sicherheitsrelevanter Fähigkeiten (Nutzungsmöglichkeit / Sicherheitsmassnahmen) |
| $Capabilities_A$ | Aktuell aktive Sicherheitsmassnahmekonfiguration |
| $Capabilities_{ALL}$ | Konfigurationsmöglichkeiten der Sicherheitsmassnahmen |
| $Capabilities_m$ | Ermittelte Sicherheitsmassnahmekonfiguration |
| *SecActivity* | Durchsetzung von Sicherheitsmassnahmen (Sicherheitsaktivität) |
| *SecAppSurface* | Anwendungsangriffsfläche |
| *SecGoal* | Bewertung der Sicherheitszielerreichung |
| *SeclevelClass* | Informationssicherheitsklassifizierung |
| *SecOperation* | Ausführung von Sicherheitsmassnahmen (interne Sicherheitsoperation) |
| Akteur | modelliert ein Typ oder eine Rolle, die ein externer Benutzer (System) einnimmt (z. B. Alice und Bob) |
| API | Programmierschnittstelle, engl. Application Programming Interface |
| Array | Datenstruktur in Form eines geordneten Felds |
| Artefakt | Schaffensergebnis im Arbeitsprozess, Informationswerte |
| Assets | Informationswerte, Datenobjekte |
| Barcode | Methode zur optischen Datenerfassung |
| Bluetooth | Funktechnik, Datenübertragung auf kleiner Distanz |
| Channels | Kommunikationskanäle |
| CRUD | Basale Operationen für die Datenverwaltung Akronym: *Create*, *Read*, *Update*, *Delete* |

| | |
|---|---|
| Exceptions | Ausnahmesituation bei Programmzuständen, beispielsweise wegen aktiven Sicherheitsmassnahmen |
| GNOME | Desktop-Umgebung mit GUI |
| GPS | Global Positioning System |
| GUI | Grafische Benutzerschnittstelle, engl. Graphical User Interface |
| HCI | Mensch-Computer-Interaktion, engl. Human Computer Interaction |
| HDMI | High Definition Multimedia Interface |
| HID | Benutzerschnittstelle, engl. Human Interface Device |
| HTTP | Hypertext Transfer Protocol (RFC2616) |
| IoT | Internet der Dinge, engl. Internet of Things |
| IPA | Intelligent Personal Assistant |
| IPC | Interprozesskommunikation, engl. Inter-process Communication |
| Methode | Objekt-orientiertes Unterprogramm |
| MVC | Software-Architekturmuster, engl. Model-View-Controller |
| NFC | Near Field Communication Funktechnik, Datenübertragung auf kurzer Distanz |
| Security Policy | Sicherheitsmodell, verortet auf dem Kontextserver |
| Socket | Kommunikationsendpunkt |
| Softwarestack | Zusammenwirkende (überlagerte) Software-Komponenten mit einem gemeinsamen Ziel |
| TCP | Transmission Control Protocol (RFC793) |
| UI | Benutzerschnittstelle, engl. User Interface |
| UML | Unified Modeling Language (ISO/IEC 19505-1) |
| URI | Uniform Resource Identifier |
| USB | Universal Serial Bus |
| W3C | World Wide Web Consortium |
| WEP | Wired Equivalent Privacy |
| Whitelist | Sicherheitsprinzip, nur explizit formuliertes wird akzeptiert |
| Widget | Element einer grafischen Benutzerschnittstelle (GUI) |

WLAN ..... Wireless Local Area Network
XML ..... Extensible Markup Language
XML-RPC ..... Extensible Markup Language Remote Procedure Call

# 1. Einleitung

Es steht außer Frage das die Nutzung von Informationstechnologien einem ständigem Wandel unterliegt. Besonders prägnant zeigt sich dies in der seit Jahren fortschreitenden Transition von stationären über leistungsfähige mobile Endgeräte hin zu vernetzten Alltagsgegenständen (auch Pervasive- und Ubiquitous-Computing genannt). Die Nutzung von mobilen Endgeräten für private Aktivitäten wie beispielsweise Kurznachrichten, Navigation, soziale Medien, Finanzaktivitäten, Fotosammlungen oder einfach nur zur Unterhaltung machen solche Geräte zu einem personalisierten Wegbegleiter. Ebenso sind mobile Endgeräte im beruflichem Umfeld nicht mehr wegzudenkenden Arbeitsmittel, da sie ein hohes potential zur Optimierung von Handlungsabläufen zur Verfügung stellen. Mobile Nutzungsszenarien sind omnipräsent.

## 1.1. Szenario

Mobile Nutzungsszenarien sind stets mit dem Zugriff auf die beruflichen und/oder persönlichen Informationen verknüpft. Solche Nutzungsszenarien werden somit zu einem attraktivem Ziel. Die Attraktivität ist dadurch gegeben, dass sensitive Daten aufgrund der wesentlich exponierteren Szenarien potentiell *leichter* abgeschöpft werden können. Folglich wird auch die Sicherheit von Informationen (Artefakten), die auf mobilen Endgeräten eingesehen, verarbeitet und übermittelt werden, immer bedeutender und dadurch entsteht ein Bedarf an neuen Sicherheitsansätzen und Modellen.

Um in mobilen Nutzungsszenarien einen geeigneten Sicherheitsdienst bereitzustellen, ist es von wesentlicher Bedeutung die situationsrepräsentierenden Kontextinformationen einzubeziehen. Solch eine Funktionalität stellt si-

cher, dass eine Kontextwahrnehmung respektive ein Kontextbewusstsein (engl. Context Awareness) dem Sicherheitsdienst zur Verfügung steht. Die resultierende Kontextsensitivität erlaubt adäquat auf das vorherrschende Nutzungsszenario einzugehen.

Die Orientierung des Sicherheitsdienstes richtet sich primär an den Artefakten aus. Diese unterliegen in der Regel unterschiedlichen Schutzzielen [DIN11, Joh09]. Diese Informationssicherheitsziele stehen im direkten *Dialog* mit den Aktivitäten eines Akteurs vermittelt durch das Medium *Endgerät*. Die Aktivität des Anwenders und des mobilen Endgerätes – im Folgenden Benutzeraktivitäten – stehen somit im Fokus des Interesses und tragen zur Bildung eines situationsrepräsentierenden Kontextbildes signifikant bei.

## 1.2. Motivation

Die in dieser Arbeit vorgenommene Auseinandersetzung geht auf die Anforderung für die Gestaltung eines kontext-sensitiven Sicherheitsdienstes ein, die Anwendungsebene als Schnittstelle zu einem Akteur und *Endpunkt* eines kontext-sensitiven Dienstes zentral einzubeziehen. Zum einen resultiert dies aus dem fundamentalen Beitrag, welcher die Benutzeraktivitäten für einen kontext-sensitiven Sicherheitsdienst leisten (s. [ADB+99]). Zum anderen aus der prinzipiellen Notwendigkeit, dass vollständige und zuverlässige kontext-sensitive Sicherheitsdienste nur mit der Kenntnis und der Hilfe der Anwendung bereitgestellt werden können (s. [CDK03, SRC84]).

## 1.3. Zielsetzung

Im Speziellen soll die Frage beantwortet werden, wie in mobilen Nutzungsszenarien die Einbeziehung der Anwendungsebene für einen kontext-sensitiven Sicherheitsdienst ausgestaltet werden kann. Das Ziel dabei ist, die Interaktion zwischen einem Akteur und einem mobilen Endgerät sicherheitsrelevant und anwendungs-spezifisch zu modellieren und einen Sicherheitsdienst ebenso auf

der Anwendungsebene anzubieten. Um Sicherheitsmassnahmen zu favorisieren trägt dieser Beitrag zu einer übergreifenden Modellierung eines sicherheitsrelevanten Kontextes bei.

## 1.4. Vorgehensweise

Kontext-sensitive Dienste weisen ein breites Spektrum auf. Der Fokus wird im Folgenden ausschliesslich auf die Einbeziehung von sicherheitsrelevanter Benutzeraktivität und der adaptiven Sicherheitsfunktionalität als Teilbereiche eines kontext-sensitive Sicherheitsdienstes liegen. Dazu wird sich deduktiv der Zielformulierung genährt und eine Konzeptualisierung der Benutzeraktivitäten vorgenommen. Die besondere Korrelation zwischen Benutzeraktivitäten und Sicherheitsmassnahmen auf der Anwendungsebene wird explizit Beachtung geschenkt.

## 1.5. Aufbau der Arbeit

Im Folgenden werden nach einem terminologischen Überblick in Kapitel 2, im Kapitel 3 ausgewählte Arbeiten vorgestellt, welche einen nicht-abschliessenden Überblick über die aktuelle wissenschaftliche Auseinandersetzung bezüglich der kontext-basierten Informationssicherheit geben. Im Rahmen der Auseinandersetzung mit der Zielformulierung wird im Kapitel 4 explizit die eingenommene Perspektive bezüglich eines Sicherheitskontextes kommuniziert. Der in dieser Arbeit verfolgte Ansatz wird im Kapitel 5 dargelegt. Der horizontale Rahmen für die Integration des vorgestellten Ansatzes wird im 6. Kapitel in Form einer Infrastruktur vorgestellt. Die vertikale Integration des vorgestellten Ansatzes wird im 7. Kapitel detailiert erläutert. Korrespondierend zu diesem 7. Kapitel sind wichtige Aspekte in Form von Quelltexten der Übersicht halber im Anhang A.1 aufgeführt und sollten nicht als optionale sondern als erforderliche Abschnitte betrachtet werden. Die Diskussion im 8. Kapitel und die Zusammenfassung im 9. Kapitel schliessen die Arbeit reflektierend ab.

# 2. Grundlagen

Im Folgenden Kapitel werden für ein besseres Verständnis der genutzten Terminologie die begrifflichen und theoretischen Grundlagen rudimentär dargelegt. Insbesondere zur Thematik der Informationssicherheit und zum Konzept des *Kontextes* werden Erläuterungen zur Verfügung gestellt. Gleichzeitig wird der Bezugsrahmen zu der in dieser Arbeit vorgenommenen Auseinandersetzung dargelegt.

## 2.1. Informationssicherheit

Die Sicherung von Informationen ist, unabhängig von den technischen Errungenschaften, eine stets begleitende Anforderung und mindestens genau so alt wie es Methoden zur Verwaltung, Kommunikation und Speicherung von Informationen gibt. In diesem Zusammenhang haben sich grundsätzliche Schutzziele herausgebildet, die der Prozess der Informationssicherheit zu adressieren hat.

### 2.1.1. Informationssicherheitsziele

Die Informationssicherheit hat primär die Aufgabe der Aufrechterhaltung folgender Informationssicherheitsziele (s. [VV13, DIN11]):

- *Vertraulichkeit* - Informationen stehen nur Personen oder Anwendungen zur Verfügung, denen einen erlaubten Zugriff gewährt wurde. Das Vermeiden eines Fremdzugriffes mag auf dem ersten Blick trivial erscheinen.

Jedoch muss die Vertraulichkeit für alle Aspekte der Informationsverarbeitung gelten und dies schliesst auch Metainformationen zu den eigentlich zu schützenden Information mit ein.

- *Integrität* - Die Unversehrtheit, Richtigkeit und Vollständigkeit von Information kann nur aufrechterhalten werden, wenn keine Manipulation von unberechtigten Personen oder Anwendungen durchgeführt werden kann. Manipulierte Informationen können in ihrer Weiternutzung zu einem Schaden führen. Daher sollte nur berechtigten Personen oder Anwendungen die Bereitstellung und Änderungen von Information gewährt werden.

- *Verfügbarkeit* - Ein fehlender Zugriff auf beispielsweise zeitkritische Informationen kann genauso wie die Nutzung manipulierter Informationen zu einem Schaden führen. Daher ist es von wesentlicher Bedeutung, dass Informationen zur Verfügung stehen, wenn der Bedarf kommuniziert wird. Im Zusammenhang mit mobilen Nutzungsszenarien respektive mobilen Endgeräten bekommt dieses Sicherheitsziel beispielsweise aufgrund von limitierter Ressourcen eine besondere Gewichtung.

Diese Informationssicherheitsziele[1] geben vor, wie ein Nutzungskontext zu bewerten ist [FK11, Joh09].

Sicherheit ist nicht gleich Sicherheit. Die in dieser Arbeit behandelte Sicherheit (engl. Security) fokussiert im wesentlich die Nutzungssituation und die durch die mobilen Nutzungsszenarien verbundenen Implikationen. Zum gegebenen Nutzungszeitpunkt wird davon ausgegangen, dass die Nutzung bestimmter Komponenten (Software, Hardware, Algorithmen, Methoden, Techniken) selbst als *sicher* (engl. Safety) betrachtet werden kann. Eine Fehlfunktion kann beispielsweise zu einem Integritätsproblem führen. Aber auch eine technisch einwandfrei funktionierende Funktion kann ein Vertraulichkeitsproblem darstellen[2]. Nichtsdestotrotz werden die relevanten Informationen die-

[1] werden in erweiterten Modellen noch zusätzlich ergänzt durch *Verbindlichkeit* und *Authentizität* als weitere Sicherheitsziele [DIN11]

[2] WLAN-Nutzung mit WEP sollte unabhänging von Informationssicherheitszielen prinzipiell im Sinne von *Safety* als unsicher betrachtet werden [FMS01]

ser Komponenten zur Verfügung gestellt und stehen somit zur Disposition. Im Folgenden wird die sich durch die mobilen Nutzungsszenarien schnell verändernde *Bedrohungslage* und die damit gefährdete Erreichung der Informationssicherheitsziele betrachtet.

### 2.1.2. Bedrohungslage

Eine Bedrohungslage stellt eine Konstellation dar, die im Sinne der betroffenen Personen potentiell zu einem unerwünschten Ereignis führen kann. Beispielsweise in dem der Schutzbedarf von Informationen nicht aufrechterhalten werden kann. Eine Bedrohungslage wird potentiell durch einen oder mehrere Schwachpunkte auf technischer-, struktureller oder organisatorischer Ebene verursacht, die zu einem Zeitpunkt an prominenter Stelle zu einem Schaden führen kann. Eine stark exponierte Lage in einem mobilen Nutzungsszenario kann beispielsweise eine Bedrohungslage darstellen. Im Rahmen der Informationssicherheit empfiehlt sich stets eine Abschätzung vorzunehmen, inwieweit die Bedrohungslage signifikant ist. Eine Risikoanalyse bei der die Wahrscheinlichkeit des Eintritts eines unerwünschten Ereignisses eingeschätzt und die daraus resultierenden Konsequenzen beziffert werden, helfen die Bedrohungslage einzuschätzen [DIN11]. Dazu geben die Informationssicherheitsziele für eine effektive Einschätzung den Schutzbedarf an. Solch eine Risikoanalyse stellt die Grundlage eines kontext-sensitiven Sicherheitsdienstes dar (Bedrohungsmodell).

## 2.2. Kontextmerkmal

Die Inklusion kontextueller Informationen für die optimierte Gestaltung von Interaktion in Informationssystemen, ist als Forschungsobjekt in letzter Zeit immer stärker in den Fokus gerückt [VM10], da der Bedarf und die Notwendigkeit *intelligentere* Informationssysteme zu entwickeln gestiegen ist. Insbesondere sind mobile Nutzungsszenarien prädestiniert für die Nutzung von Kontextinformationen.

Trotz eines *stillschweigenden* Konsens existiert kein *expliziter* Konsens über das was *Kontext* ist und was nicht. Dies ist unter anderem der Tatsache geschuldet, dass das Kontextkonzept in verschiedenen wissenschaftlichen Disziplinen an Bedeutung gewonnen hat und somit der jeweilige Standpunkt und der Fokus bezüglich des Kontextes ebenso verschiedene Definitionsausprägungen hervorbringt. Erschwerend kommen die Schwierigkeiten bei der Konzeptualisierung von *Kontext* hinzu. In [BB05] werden beispielsweise 150 Kontextdefinitionen für eine Metananalyse herangezogen.

Eine etymologische Definition ist genauso wie eine disziplin-übergreifende Definition nicht zielführend, da die Abstraktionsebene rekursiv in die Definitionsableitung einwirkt. Gerade weil dieser Umstand die Kommunikation erschwert, muss trotz der Anstrengungen eine einheitliche und allgemein akzeptierte Begriffsdefinition hervorzubringen (s. [BB05]), im Rahmen eines Diskurses über die Konzeption eines kontext-sensitiven Sicherheitsdienstes die eingenommene Perspektive kommuniziert werden. Im Umfeld der Mensch-Computer-Interaktion (HCI) hat der Begriff Kontext beispielsweise eine aufzählende Definition:

> *„Nutzungskontext: Die Benutzer, Arbeitsaufgaben, Arbeitsmittel (Hardware, Software und Materialien) sowie die physische und soziale Umgebung, in der das Produkt genutzt wird.“* [DIN98]

Eine aufzählende Herangehensweise schränkt allerdings gleichzeitig die Flexibilität ein. Da nicht davon ausgegangen werden kann, dass die aufgezählten Aspekte für alle Situationen gleichermassen eine Gültigkeit haben (s. [ADB+99]). In der Forschungsdisziplin des *Context-Aware Computing* hat sich daher folgende Definition von Dey und Abowd etabliert:

> *„Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves.“* [ADB+99]

Die Ausrichtung dieser Definition auf eine Situation erlaubt die Nutzung verschiedener Informationen, solange diese die Situation beschreiben. Dies erhöht

erheblich die Anwendbarkeit dieser Definition beispielsweise bei der Anwendungsentwicklung. In [ADB+99] geben Dey und Abowd mit der *Location-*, *Identity-*, *Activity-*, und *Time*-Dimension ebenso eine Empfehlung ab, in welchen Dimensionen besonders signifikante Kontextinformationen zu ermitteln sind. Im Hinblick auf die Gestaltung eines kontext-sensitiven Sicherheitsdienstes leistet die *Activity*-Dimension einen besonderen Beitrag. Dies wird von Dey und Abowd zusätzlich unterstrichen:

> *„Aktivität, (...) beantwortet die grundsätzliche Frage; was sich denn gerade in der Situation ereignet."* [ADB+99, (Übers. vom Verf.)]

### 2.2.1. Kontextbewusstsein

Beeinflusst eine Kontextinformation das Verhalten eines Systems oder einer Anwendung, so wird von der Eigenschaft der Kontextwahrnehmung respektive des Kontextbewusstseins (engl. Context Awareness) einer Komponente gesprochen. In [ST94] beschreiben Schilit und Theimer solch eine Kontextsensitivität wie folgt:

> *„Context-aware computing is the ability of a mobile user's applications to discover and react to changes in the environment they are situated in."*

und führen weiter aus

> *„ (...) to adapt according to its location of use, the collection of nearby people and objects, as well as the changes to those objects over time."* [ST94]

Während Schilit und Theimer Kontextbewusstsein als eine Funktion von Kontextentitäten betrachtet, wird von Dey und Abowd ein Kontextbewusstsein eines Systems oder Anwendung als eine Funktion der Benutzeraktivität betrachtet:

> *„A system is context-aware if it uses context to provide relevant information and/or services to the user, where relevancy depends on the user's task.“* [ADB+99]

Solch ein Ausrichtung auf die Benutzeraktivität unterstützt die angemessenere Bereitstellung von Diensten insbesondere von Sicherheitsdiensten.

## 2.3. Adaptivität

Die Konzeption von Informationssystemen kann die Kontextwahrnehmungsfähigkeit auf unterschiedliche Weise einsetzen. Prinzipiell wird eine *Adaptivität* (engl. Adaptivity) als eine Anpassung des Verhaltens (Reaktion) vonseiten des Informationssystems selbst verstanden. Dies ist klar abzugrenzen von einer *Adaptierbarkeit* (engl. Adaptability) die primär eine Anpassung (Individualisierbarkeit) vonseiten der Akteure bezeichnet. Anwendungen, die eine im Hintergrund laufende kontext-sensitive Adaptivität unterstützen, stellen einem Akteur einen angemesseneren Dienst zu Verfügung. Die Kontextwahrnehmungsfähigkeit muss nicht zwangsläufig zu einer Adaption der Anwendungsfunktionalität führen. In [DAS01] kategorisiert Dey kontext-sensitive Eigenschaften von Anwendungen wie folgt:

- *Präsentation* - Anwendungseigenschaft dem Benutzer Kontextinformationen zur Verfügung zu stellen (beispielsweise geografische Ortsangaben)
- *Ausführung* - Anwendungseigenschaft die eigene Funktionalität automatisch und flexibel für den Benutzer zu adaptieren
- *Annotation* - Anwendungseigenschaft Informationen mit Kontextinformationen anzureichern (beispielsweise Annotation von Bildern)

Ein kontext-sensitiver Sicherheitsdienst wird primär in die Kategorie *Ausführung* eingeordnet, da es eine sicherheitsrelevante Anwendungsfunktionalität zur Verfügung stellt. Diese Sicherheitsfunktionalität sollte nach Möglichkeit automatisch, transparent und zeitnah im Bedarfsfall zum Einsatz kommen.

Solch eine kontext-sensitive Sicherheitsadaption wird allerdings auch ein nichtdeterministisches Verhalten der Anwendung nach sich ziehen. Hierzu ergeben sich weitere Anforderungen für die Interaktionsgestaltung [HM06, DIN06], wodurch auch kontext-sensitive Eigenschaften wie die der *Präsentation* ein weiteres Merkmal eines Sicherheitsdienstes darstellt.

## 2.4. Zusammenfassung

In den vorangegangenen Abschnitten wird ein terminologischer Überblick zur Verfügung gestellt, um zu einem besseren Verständnis beizutragen. Es werden die Informationssicherheitsziele erläutert und eine klare Abgrenzung zur Funktionssicherheit vorgenommen. Ebenso wird der Zusammenhang zwischen Bedrohungslage und mobilen Nutzungsszenarien explizit erörtert. Des Weiteren wird der *Kontext* konzeptuell erfasst und der Bezug zur Benutzeraktivität aufgezeigt. Mit den abschliessenden Ausführungen zur Adaptivität werden Eigenschaften kontext-sensitiver Adaption vorgestellt.

Im Folgenden wird die vorgestellte und in der Forschungsdisziplin *Context-Aware Computing* allgemein-akzeptierte Kontextdefinition von Dey und Abowd [ADB+99] als Leitlinie für die weitere Auseinandersetzung dienen.

In den nächsten zwei Kapiteln soll die in dieser Arbeit vorgenommene Auseinandersetzung in den aktuellen wissenschaftlichen Diskurs bezüglich kontext-basierter Informationssicherheit gestellt werden.

# 3. Diskurseinordnung

Die wissenschaftliche Auseinandersetzung mit der Thematik der kontext-sensitiven Sicherheitsadaption findet seit einigen Jahren verstärkt statt. Dies ist auch der Tatsache geschuldet, dass die Informationstechnologien sich rasant weiterentwickelt haben, wodurch neue Nutzungsszenarien entstehen und diese wiederum neue Sicherheitsansätze und Modelle erfordern.

Prinzipiell impliziert die Auseinandersetzung mit kontext-adaptiven Systemen zwei Schwerpunkte: zum einen die Kontextbehandlung und zum anderen die Adaptivität. Bei der Sicherheitsadaption fällt die starke Konzentration auf eine adaptive Zugriffskontrolle auf (s. [JGK14, JAB10, MB03, CFZA02]). Hier ist eine stärkere unterstützende Adaptivität im Sinne eines Sicherheitsservices und nicht eines Sicherheitsmechanismus wünschenswert. Hierzu zeigen die Zugriffskontrollmodelle selbst Weiterentwicklungen auf [SP03], die wiederum in die Sicherheitsadaptionmodelle einfliessen [CCB08, BGF+10]. In [BGF+10] wird ein kontext-sensitives Nutzungsrechtemodell vorgestellt, dass eine Zugriffsentscheidung nicht ausschliesslich auf statische Berechtigungseinstellungen der Informationswerte und Nutzer trifft, sondern zusätzlich die Autorisierung anhand von Konditionen und der Erfüllung von Verpflichtungen (Obligation) bewertet. Ersteres korrespondiert mit dem Nutzungskontext und letzteres können anwendungskontextuelle Anforderungen repräsentieren. Solche Sicherheitsdienste müssen bis hin zur Anwendungsebene implementiert sein [JAB10, BPVG09].

Die Kontextbehandlung wiederum zeigt ganz eigene Anforderungen auf. Je nach betrachteten Anwendungsfall unterscheiden sich die genutzten Kontextdimensionen. Um die erschöpfende Auswahl an Informationen zu verwalten empfehlen Villegas und Müller eine Einordnung und stellen dazu in [VM10] eine Kontext-Taxonomie vor, in der die Aktivitätsdimension allerdings keine

weitere Unterteilung erfährt. Die kontinuierliche Überwachung des Kontextes wird in [EPS10] thematisiert, während dies wiederum aufgrund der Ressourcennutzung abgewogen werden sollte [FKRL09]. Bezüglich der unterschiedlichen Kontextquellen wird die Anwendungsebene in [BPVG09] mit Blick auf die Interaktion explizit modelliert und in [Joh09] erlauben die vorgestellten Sicherheitsziele eine Ableitung der relevanten Aktivitäten. In [EPS10] empfehlen die Autoren eine Typisierung der in die Aktivitäten einbezogenen Artefakte vorzunehmen.

Kontext-sensitive Systeme und im speziellen das kontext-adaptive Sicherheitsverhalten von mobilen Endgeräten respektive Anwendungen basiert auf komplexen Modellen und Konzepten [YDM12], welche die gegebene reale Komplexität der Thematik abzubilden versucht. Um der Komplexität Herr zu werden, werden im wissenschaftlichem Diskurs stets einzelne Aspekte betont behandelt oder allgemeine abstrakt-konzeptuelle Modelle entwickelt. Desweiteren zeigen aktuelle Arbeiten eine Konzentrationsverschiebung auf die Kontextverwaltung respektive auf die Behandlung des Lebenszyklus der Kontextinformationen [VM10]. Die Kernherausforderungen stellen die Aggregation und Abstraktion (Modellierung) sowie die Interpretation (Inferenz) von Kontextinformationen dar [YDM12, FK11, BBH$^+$09, BDR07]. Trotz verschiedener interessanter Ansätze ([MATA$^+$14, Kie12]) bekommen ontologie-basierte Kontextmodelle aufgrund ihrer Expressivität und der Möglichkeit Informationen abzuleiten eine signifikante Rolle bei der Entwicklung kontext-sensitiver Systeme [BBH$^+$09, BDR07]. Hierzu müssen für eine zeitkritische Sicherheitsadaption die erhöhten Rechenkosten in die Modellierung einbezogen werden (s. [Ay07]).

In dieser Arbeit liegt der Schwerpunkt auf der Kontextermittlung (Context Sensing) in Benutzeranwendungen (Anwendungskontext) und dem kontextadaptivem Anwendungsverhalten, wobei die Sicherheitsrelevanz hierbei modellbildend ist. Dazu wird im nächsten Kapitel die eingenommene Perspektive bezüglich eines sicherheitsrelevanten Kontextes vorgestellt.

# 4. Sicherheitsrelevanter Kontext

Sicherheitsrelevanter Kontext ergibt sich in erster Linie durch die betroffenen Sicherheitsziele und durch die Kontexte relevanter Entitäten. Wobei in Anlehnung an der akzeptierten allgemeinen Kontextdefinition von Dey et al. [ADB+99] die *Entitäten* als abstraktes Konzept für die verschiedenen Objekte oder Teilnehmenden einer Interaktion betrachtet werden können (s. [Joh09]). Mit Bezug auf die Sicherheitsrelevanz erhält der Kontext, als der nutzungseinbettende, konzeptuelle und operationelle Raum, eine weitere Gewichtung (Sicherheitsdimension), die in der wissenschaftlichen Auseinandersetzung zu verschiedenen Definition führt. Die jeweiligen Perspektiven der Autoren (s. [MB03, ABKS09, Joh09, BPVG09, JGK14]) kommen dabei in den Definitionen zum Ausdruck. Es werden Entitäten in den Mittelpunkt ([ABKS09]) gestellt, konkrete Sicherheitsbedrohungen und Ziele vorgegeben ([Joh09]), oder wie in der folgenden Definition, die potentiellen Informationsbereiche a priori festgelegt:

> *„A security context is a set of information collected from the users environment and the application environment and that is relevant to the security infrastructure of both the user and the application.“* [MB03]

Eine aufzählende Herangehensweise erweitert den Einzugsbereich für potentielle Informationen, schränkt allerdings gleichzeitig die Flexibilität im Sinne einer Allgemeingültigkeit ein:

> *„A security context is any information that can be used to characterize the security situation of a communication environment. The security situation can be characterized in terms of possible security threats, user status*

> *and requirements with respect to information protection, available communication and computing resources which can be exploited by a system in order to specify and enforce proper actions to guarantee end-to-end security.“* [BPVG09]

Viel wichtiger wird durch die Sicherheitsgewichtung der Umstand, dass die Informationssicherheit keinen Zustand, sondern einen kontinuierlichen Prozess darstellt, der sich generell und insbesondere in mobilen Nutzungsszenarien wiederholt an die verändernden Bedingungen anpasst. Darüber hinaus ist die selektive gegenüber einer erschöpfenden Informationsauswahl zu adressieren. Dazu folgende Definition, die sich an der Kontextdefinition von Dey et al. [ADB+99] orientiert:

> *„Security context is a set of contextual information considered relevant for the process of security, regarding a particular task or activity. Contextual information is any information that can be used to characterize the situation of an entity. An entity can be a person or a device, (. . . )“* [JGK14]

Diese Definition von Jovanovikj et al. beschränkt sich nicht auf bestimmte Informationsbereiche. Durch die Ausrichtung auf einen Sicherheitsprozess wird die Sicherheitsdimension adäquat adressiert. Diese Prozessperspektive bezieht ein grösseres Zeitfenster ein, als das es von einer *Situation* implizit kommuniziert werden könnte. Gleichzeitig beschränkt es sich auf die *Aktivität*, die als Bezugspunkt die Sicherheitsanforderung *Schutzbedarf* vorgibt.

Unter Sicherheit wird die Informationssicherheit mit den Sicherheitszielen Vertraulichkeit, Integrität und Verfügbarkeit [DIN08] verstanden und bezüglich des Sicherheitskontextes wird die Perspektive von Jovanovikj et al. [JGK14] als Leitlinie für die weitere Auseinandersetzung dienen. Im nächsten Kapitel wird dazu der konzeptuelle Ansatz vorgestellt.

# 5. Konzeptuelles Modell

Im Rahmen einer Betrachtung der Benutzeraktivität mit dem Schwerpunkt Informationssicherheit, gilt zu beachten, dass die zusätzliche Sicherheitsdimension konkrete Ziele vorgibt und sich somit in Interaktion mit dem vorherrschenden Nutzungskontext stellt. Eine kontext-sensitive Adaption wird somit um die Sicherheitsdimension herum zentriert. Dazu soll folgende Herangehensweise vorgestellt werden.

## 5.1. Methodische Überlegungen

Um in die Lage zu gelangen Benutzeraktivitäten und deren Sicherheitsrelevanz modellieren zu können, muss deren rekurrente Struktur identifiziert werden. Methodisch betrachtet kann zunächst festgehalten werden, dass die Handlungsmöglichkeiten eines Benutzers auf die Nutzungsmöglichkeiten des mobilen Endgeräts beschränkt sind. Solch eine innere Gliederung muss überdies auch einen Zugang für die Bewertung der Sicherheitsrelevanz bieten. Ein zielführender Ansatz stellt das Konzept der *Angriffsfläche* [MW11] von Manadhata und Wing dar. Eine Angriffsfläche kann demnach als ein Satz von Interaktionsmerkmalen gesehen werden, die potentiell den Informationssicherheitszielen gegenüber stehen. Dieses Konzept wird rudimentär im Kontextmodel von Jovanovikj et al. [JGK14] bereits einbezogen. Im Sinne einer kontext-sensitiven Adaption findet es allerdings keine weitere Verwendung.

Der Ansatz von Manadhata und Wing [MW11] stellt die nötige Granularität zur Verfügung, um auf Anwendungsebene die Benutzeraktivitäten und deren Sicherheitsrelevanz methodisch zugänglich zu machen (Context Sensing). Zusätzlich erlaubt das Konzept der Angriffsfläche [MW11], den zentralen Bereich

der kontext-sensitiven Sicherheitsadaption konzeptuell zu erfassen (Context-aware Adaption). Daher wird die weitere Auseinandersetzung sich daran orientieren und entsprechend der Zielsetzung eine weiterführende Modellierung vornehmen.

## 5.2. Modellierung

### 5.2.1. Angriffsfläche

Nach Manadhata und Wing [MW11] stellt eine Angriffsfläche einen Satz an Merkmalen dar, die potentiell für einen Angriff genutzt werden. Zu diesen Merkmalen zählen die Methoden (*Methods*) die angewendet werden können und die Daten (*Data*). Zusätzlich werden die Eintritts- und Austrittspunkte identifiziert. Letzteres stellen nutzbare Kanäle (*Channels*) dar, über die eine Kompromittierung statt finden kann (*Attacks*). In [MW11] werden diese Merkmale als Ressourcen bezeichnet und dementsprechend eine Systemangriffsfläche anhand der Systemressourcen wie in Abbildung 5.1 definiert.

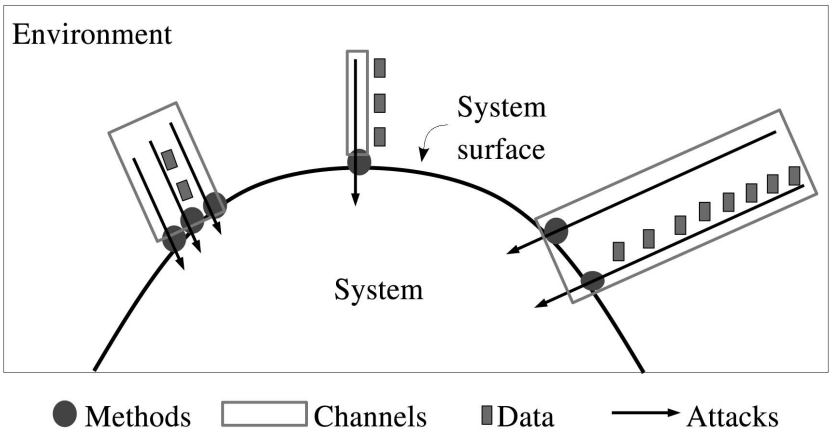


Abb. 5.1.: Eine Systemangriffsfläche ist die Teilmenge der Systemressourcen (Methods, Channels, Data) [MW11].

Das primär in [MW11] betrachtete Sicherheitsszenario geht von einer Manipulation der Ressource *Data* aus. Weitere Sicherheitsszenarien sind zusätzlich

möglich. Beispielsweise kann die Ressource *Channel* Implikationen einbeziehen und somit die Integrität respektive die Vertraulichkeit der Informationen, die darüber fliessen verletzen (Security Incident).

Für die Anwendungsebene, auf der Benutzeraktivitäten in der Regel verortet sind, können entsprechend analog zur Abbildung 5.1 folgende Interaktionsmerkmale identifiziert werden: die angebotenen Anwendungsoperationen $(Operations)$ die auf Datenobjekte[1] $(Assets)$ über verschiedene Kommunikationskanäle $(Channels)$ angewendet und vom Benutzer direkt[2] oder indirekt[3] durchgeführt werden können. Somit kann für eine Anwendungsangriffsfläche $(SecAppSurface)$ folgendes festgehalten werden:

$$SecAppSurface \subsetneq Assets \cup Operations \cup Channels \tag{5.1}$$

Nicht jede Anwendungsoperation, jedes Datenobjekt oder jeder Kommunikationskanal trägt zur Anwendungsangriffsfläche bei. Die Interaktionsmerkmale die allerdings dazu beitragen sind sicherheitsrelevant und konstituieren den Raum sicherheitsrelevanter Benutzeraktivitäten (s. [JAB10, ADB+99]).

### 5.2.2. Sicherheitsmassnahmen

Sicherheitsmassnahmen minimieren die Wahrscheinlichkeit des Eintritts eines unerwünschten Ereignis [DIN11]. Dies geht einher mit der Identifizierung der Bedrohungslage (s. Kap. 2.1.2), der Einschätzung des von der Bedrohungslage ausgehenden Risikos (Wahrscheinlichkeit), und der Ausführung und Durchsetzung von Sicherheitsmassnahmen. Diese Aufzählung impliziert bereits, dass die Informationssicherheit in mobilen Nutzungsszenarien stets unbeständig ist. Ein kontinuierlicher Prozess der wiederholt die Sicherheitsmassnahmen an die verändernden Nutzungsszenarien adaptiert ist somit essenziell (s. [FK11, Joh09]).

Die Informationssicherheit wird durch Sicherheitsmassnahmen sichergestellt. Eine Möglichkeit dieses Ziel zu erreichen ist unter anderem die Reduzierung

[1] Datenobjekte repräsentieren die schützenswerten Informationen
[2] Beispiele für Eingabeschnittstellen (HID): Touch, Pointer, Keyboard, Voice-input
[3] Vorschaufähigkeiten der Anwendung (Auflistung, Teaser, Aggregation)

der Anwendungsangriffsfläche (5.1) (s. [MW11]). Für eine Sicherheitsmassnahme in Form einer Angriffsflächenreduzierung ($SecAppSurface_{k_{t_2}}$) getriggert durch eine Änderung des Sicherheitskontextes gilt demnach:

$$|SecAppSurface_{k_{t_2}}| < |SecAppSurface_{k_{t_1}}| \tag{5.2}$$

Dies erlaubt unerwünschte Ereignisse von vornherein auszuschliessen, da eine Angriffsflächenreduzierung entsprechend (5.1) zu einer Einschränkung der Operationsmöglichkeiten, des Zugriffs auf Informationswerte[4] oder der Kommunikationskanalnutzung führt. Dieser Ansatz reduziert die Nutzungsmöglichkeit als Sicherheitsmassnahme und soll hier weiter verfolgt werden. Wie solch eine Nutzungsmöglichkeit gesteuert werden soll, wird im nächsten Abschnitt erläutert.

### 5.2.3. Nutzungsmöglichkeit

Eine Anwendung bietet eine Nutzungsmöglichkeit an, die als Handlungsmöglichkeit[5] vom Benutzer wahrgenommen wird. Diese Nutzungsmöglichkeit gibt den Rahmen für die Benutzeraktivitäten vor. Die sicherheitsrelevante Teilmenge dieser Nutzungsmöglichkeit stellt die Anwendungsangriffsfläche dar (s. Kap. 5.2.1).

Neben den Anwendungsoperationen die in ihrer Summe die Nutzungsmöglichkeit bilden, kommen weitere Operationen hinzu, die die Sicherheitsmassnahmen integrieren. Diese Sicherheitsoperationen ($SecOperations$) stellen beispielsweise die Durchsetzung der Angriffsflächensteuerung sicher. Um eine kontext-sensitive und flexible Weise der Adaption vornehmen zu können, werden die Sicherheitsoperationen mit dem sicherheitsrelevanten Teil der Nutzungsmöglichkeit — der Anwendungsangriffsfläche (SecAppSurface) — wie folgt gekapselt:

$$Capabilities \subseteq SecAppSurface \cup SecOperation \tag{5.3}$$

[4] ff. *Assets* für Informationswerte, Datenobjekte

[5] siehe [Nor99] zur Theorie der sogenannten *Affordance* für grafische Benutzerschnittstellen

Für die Adaption wird allerdings strukturelles Wissen benötigt, dass sich je nach Anwendung spezifisch unterscheidet. Daher soll dieses Wissen (Regelwerk) allgemein als anwendungs-spezifische Sicherheitskonfiguration ($SecConfCaps$) definiert werden und zusammen mit den sicherheitsrelevanten Anwendungsfähigkeiten ($Capabilities$) unter Sicherheitsaktivitäten gekoppelt werden. Die Sicherheitsaktivitäten ($SecActivity$) werden als Tupel wie folgt repräsentiert:

$$SecActivity = (Capabilities, SecConfCaps) \tag{5.4}$$

Die Sicherheitsaktivitäten sind kein Teil der Anwendungsangriffsfläche. Sondern adaptieren Letzteres mittels *interner* Sicherheitsoperationen ($SecOperations$). Die anwendungs-spezifische Sicherheitskonfiguration ($SecConfCaps$) definiert die Art und Weise dieser Sicherheitsoperationen für die Durchsetzung der Sicherheitsmassnahmen.

Die $Capabilities$ stellen sicherheitsrelevante Fähigkeiten der Anwendungen dar. Sowohl im Sinne der Nutzungsmöglichkeit als auch der Massnahmen, die diese Nutzungsmöglichkeit auf Anwendungsebene steuern. Im nächsten Abschnitt soll unter Verwendung der $Capabilities$ die Adaptivität der Informationssicherheit näher beleuchtet werden.

### 5.2.4. Adaption

Die Sicherheitsmassnahmen sollen das potenzielle Sicherheitsrisiko minimieren, dass durch den vorherrschenden Nutzungskontext hervorgerufen wird (s. Kap. 5.2.2). Die Anwendungsebene stellt sicher, dass die Sicherheitsmassnahmen in Form von $Capabilities$ entsprechend mittels $SecActivity$ umgesetzt werden (s. Kap. 5.2.3).

Die Erreichung der Informationssicherheitsziele ist eine Funktion der Sicherheitsmassnahmen. Die Informationssicherheitsziele (s. Kap. 2.1.1) sind als Merkmale ($SecLevel_{Asset}$) der Assets zu betrachten. Für die Klassifizierung soll eine hierarchische Einteilung ($SecLevelClass$) der Übersicht halber angenommen werden. Wobei die Wertigkeit mit den Informationssicherheitsanforderung kor-

reliert.

$$SecLevel_{Asset} \in SeclevelClass \tag{5.5}$$

Die adaptive Informationssicherheit wird durch die intentionale Veränderung des Sicherheitskontextes erreicht, indem der sicherheitsrelevante Teil der Nutzungsmöglichkeit adaptiert wird. Sei dazu der zu ermittelnde Anwendungskontext $AppContext$ einer Benutzeraktivität (lokal) während eines invarianten Nutzungskontextes (global) gegeben durch ein Tupel von Tupels mit

$$AppContext^{6} = \big((App), (Asset), (Operation), (Channel)\big) \tag{5.6}$$

und dem Sicherheitskontext (s. Kap. 4) mit

$$Seccon = \big(\dots, (Place), (AppEnv), (AppContext), (Time), \dots\big). \tag{5.7}$$

In dem Beispiel ist der Sicherheitskontext vorwiegend durch den Anwendungskontext der Benutzeraktivität[7] gegeben. Somit kann für eine spezifische Benutzeraktivität mit $Operation_u$ ein entsprechender Sicherheitskontext ermittelt werden. Die Operation $Operation_u$ wird in einer Anwendung $App_u$ auf ein Objekt $Asset_u$ mit entsprechenden Sicherheitsziel $SecLevel_{Asset_u}$ mit $SecLevel_{Asset_u} \in SeclevelClass$ durchgeführt. Dabei ist die Anwendung $App_u$ hinsichtlich der *aktiven* Sicherheitsmassnahmen respektive der Anwendungsfähigkeiten $(Capabilities)$ in einem Ausgangszustand $Capabilities_A$ mit $Capabilities_A \subseteq Capabilities_{App_u}$. Aus Gründen der Verdeutlichung ist der Sicherheitskontext $Seccon_u$ wie folgt reduziert wiedergegeben:

$$Seccon_u = \Bigg(\dots, \underbrace{\Big(\overbrace{(Capabilities_A, \dots)}^{App_u}, \overbrace{(SecLevel, \dots)}^{Asset_u}, \overbrace{(\dots)}^{Operat._u}, \dots\Big)}_{AppContext_u}, \dots\Bigg) \tag{5.8}$$

Die Bewertung der Zielerreichung $(SecGoal)$ anhand des Sicherheitskontex-

[6] in Anlehnung an [JGK14], wobei ein verändertes Modell zugrunde gelegt wird

[7] exemplarisch sind die geografische und die zeitliche Dimension, und der anwendungsübergreifende Ausführungskontext (s. Kap. 6.1.2) zusätzlich aber nicht abschliessend aufgelistet

tes ist gegeben durch:

$$SecGoal_u : (SecLevel_{Asset_u} \Rightarrow Seccon_u) \rightarrow \{0, 1\} \tag{5.9}$$

Genügt der Sicherheitskontext $Seccon_u$ dem Informationssicherheitsziel $SecLevel_{Asset}$ des Objektes $Asset_u$, so kann die Benutzeraktivität mit $Operation_u$ durchgeführt werden. Dies impliziert, dass die *aktiven* Sicherheitsmassnahmen[8] mittels $Capabilities_A$ zusammen mit den verschiedenen Anteilen des Sicherheitskontextes den Beitrag für solch eine Entscheidung leistet. Da eine Sicherheitsmassnahme in ihrer maximalen Ausprägung ohnehin nur Operationen zur Verfügung stellt, die die Bewertung *SecGoal* (5.9) *positiv* durchlaufen, besteht der Bedarf eines Kriteriums, dass eine flexible und kontext-sensitive Adaption ermöglicht (s. [Joh09]). Dazu soll der Sicherheitskontext $(Seccon)$ selbst einer Bewertung unterworfen werden. Hierfür wird eine Abbildung auf die Sicherheitsklassifizierung vorgenommen, die bereits für die Datenobjekte (Assets) verwendet wird $(SecLevelClass)$. Die Bewertung des Sicherheitskontextes $(AssessSeccon)$ ist gegeben durch:

$$AssessSeccon_u : Seccon_u \rightarrow SeclevelClass \tag{5.10}$$

Die Bewertung der Zielerreichung $(SecGoal)$ erfolgt *nicht* mehr direkt über den Sicherheitskontext $Seccon_u$ (5.9), sondern über die Sicherheitskontextbewertung *AssessSeccon* (5.10). Dazu wird folgende Unterscheidung vorgenommen:

$$SecGoal(SecLevel_{Asset_u})^9 = \begin{cases} 0, & SecLevel_{Asset_u} > AssessSeccon_u \\ 1, & SecLevel_{Asset_u} \approx AssessSeccon_u \\ 0, & SecLevel_{Asset_u} < AssessSeccon_u \end{cases} \tag{5.11}$$

In der Regel wird Aufgrund der Mehrdimensionalität des Sicherheitskontextes $Seccon_u$, eine Gleichwertigkeit der Sicherheitsklassifizierung nur selten ein-

[8] dies kann das Nichtvorhandensein von Massnahmen als auch Massnahmen in ihrer maximalen Ausprägung bedeuten (s. (5.3) in Kap. 5.2.3 auf Seite 18)

[9] mit $\{AssessSeccon_u, SecLevel_{Asset_u}\} \in SecLevelClass$

treten. Es empfiehlt sich daher eine akzeptable Näherung mittels spezifischer Sicherheitsrichtlinie (Security Policy) zentral festzulegen.

Mit der Unterscheidung (5.11) ist die kontext-sensitive und flexible Adaption möglich, indem mittels Sicherheitsmassnahmen respektive der Anwendungsfähigkeiten $(Capabilities)$ der Sicherheitskontext $Seccon_u$ in ein Gleichgewicht mit den Informationssicherheitszielen $SecLevel_{Asset}$ gebracht wird. Wie auf Seite 20 aufgeführt folgt aus (5.8), dass die Anwendungsfähigkeiten $(Capabilities)$ ein Bestandteil des Sicherheitskontextes sind. Tragen die entsprechend effektiven Sicherheitsmassnahmen nicht ausreichend zur Zielerreichung bei (erster Fall in (5.11)), respektive sind die Sicherheitsmassnahmen für den aktuellen Nutzungskontext zu restriktiv (dritter Fall in (5.11)), dann schliesst sich daran eine Adaptionsermittlung $(SecAdaption)$ an. Hierzu wird die Sicherheitsmassnahme ermittelt, die zu einem Sicherheitskontext $Seccon_u$ führt[10], der wiederum den Informationssicherheitszielen $SecLevel_{Asset}$ des Objektes $Asset_u$ genügt. Der gegebene Sicherheitskontext $(Seccon_u)$ beinhaltet dazu bereits die Zielvorgabe $(SecLevel_{Asset})$ und den Ausgangszustand $(Capabilities_A)$ (s. (5.8)). Die für die Adaption genutzte Sicherheitsmassnahme in Form von $Capabilities_m$ mit $Capabilities_m \subseteq Capabilities_{App_u}$ wird wie folgt abgeleitet:

$$SecAdaption_u : Seccon_u \rightarrow Capabilities_m \tag{5.12}$$

Im ersten Fall aus (5.11) würde die Anwendungsfähigkeit $Capabilities_m$ bezüglich des Umfangs im Vergleich zu $Capabilities_A$ kleiner ausfallen. Um somit eine Adaption zu einem höherwertigen Sicherheitskontext $Seccon_u$ durchzusetzen. Im dritten Fall (5.11) ist das Informationssicherheitsziel $SecLevel_{Asset_u}$ bereits erreicht. Dennoch würde eine Adaption ermittelt werden, die eine Anpassung der Anwendungsfähigkeiten vornimmt, um die nicht zwingend notwendigen Sicherheitsmassnahmen zu reduzieren. Somit würde die Anwendungsfähigkeit $Capabilities_m$ bezüglich des Umfangs grösser ausfallen im Vergleich zu $Capabilities_A$, um eine Adaption zu einem minderwertigen Sicherheitskontext $Seccon_u$ vorzunehmen, dass dennoch die Informationssicherheit des Objektes $Asset_u$ mit $SecLevel_{Asset}$ aufrechterhält. Auf der Ebene der Benutzeraktivität bedeutet das für die *angeforderte* Operation, dass die $Operation_u$ nur durch-

[10] beispielsweise nach dem „Teile-und-herrsche“-Prinzip

geführt werden kann, wenn die Operation nach dem Adaptionsprozess mit $Operation_u \in Capabilities_m$ weiterhin zur Verfügung steht. Die Sicherheitsaktivität (5.4), die für die Durchsetzung der Adaption verantwortlich ist, ist ebenso für die Behandlung solcher Ausnahmefälle verantwortlich.

Der Adaptionsprozess besteht aus der Ermittlung (*SecAdaption* (5.12)) und der Durchsetzung der Sicherheitsmassnahmen (*SecActivity* (5.4)). Er gewährleistet die Informationssicherheit, indem er der Bedingung $SecLevel_{Asset_u} \approx AssessSeccon$ aus (5.11) nachkommt. Im Extremfall mit dem kleinsten Satz an Anwendungsfähigkeiten. Gibt der Sicherheitskontext $(Seccon_u)$ die Möglichkeit solch einer Bewertung (5.11) nicht her, so nimmt die aktuelle Benutzeraktivität einen Ausnahmeverlauf (Exception). Die flexiblen Sicherheitsmassnahmen werden in Form von Anwendungskonfigurationen (*Capabilities*) kontextsensitiv bereitgestellt, welche die Fähigkeiten der Anwendung und somit die möglichen Benutzeraktivitäten regulieren.

## 5.3. Zusammenfassung

In den vorangegangenen Abschnitten werden einleitend methodische Überlegungen vorgestellt, um sich der Modellierung sicherheitrelevanter Benutzeraktivitäten zu nähern. Auf Basis des Ansatzes von Manadhata und Wing [MW11] werden analog die Interaktionsmerkmale einer Benutzeraktivität identifiziert und als Anwendungsangriffsfläche vorgestellt. Aufgrund der Informationssicherheitsziele korrelieren die sicherheitsrelevanten Benutzeraktivitäten mit dem kontext-adaptiven Sicherheitsverhalten der Benutzeranwendungen. Dementsprechend wird zusätzlich die kontext-sensitive Sicherheitsadaption konzeptuell erfasst. In Anlehnung an [MW11] wird dazu das dynamische Adaptionsverhalten als Angriffsflächenreduzierung konzeptualisiert vorgestellt.

Im nächsten Kapitel soll ein infrastrukturelles Modell für die Verarbeitung von Kontextinformationen vorgestellt werden, das den Schwerpunkt auf die sicherheitrelevanten Benutzeraktivitäten als Kontextquelle respektive dem Anwendungskontext, inklusive der Ableitung von Sicherheitsmassnahmen legt.

# 6. Architektur

Das kontext-adaptive Sicherheitsverhalten von Systemen respektive Anwendungen basiert auf einer Infrastruktur die einen *Lebenszyklus* der Kontextinformation im besten Fall vollständig unterstützt (engl. Context Provisioning). Solch ein Lebenszyklus erstreckt sich von der Ermittlung, Aggregation und Abstraktion der Kontextinformationen, über die Interpretation, das Monitoring und der Verwaltung, bis hin zur Löschung von obsoleten Kontextinformationen [VM10]. Somit stellt die Modellierung sicherheitsrelevanter Benutzeraktivitäten als Kontextquelle einen vertikalen Ausschnitt dar. In diesem Kapitel wird dieser vertikale Ausschnitt in seiner Einbettung in eine potenziellen Architektur, entlang der Architekturkomponenten weiter vorgestellt.

## 6.1. Architekturkomponenten

Für den Entwurf einer Architektur ist eine allgemein anerkannte Vorgehensweise die, dass für jeden Gegenstand der Betrachtung eine korrespondierende Komponente bereitgestellt wird. Solch eine Entkopplung ist mit Blick auf die Konzeption kontext-sensitiver Systeme besonders zielführend [DAS01], da es die Komplexität reduziert und die Wiederverwendbarkeit von Komponenten unterstützt (s. [BDR07, CDK03]). Die folgende Einteilung ergibt sich bereits aus der vorgenommen Modellierung:

- Anwendungsspezifische Benutzeraktivität $(AppContext)$
- Sicherheitskontext-Ermittlung $(Seccon)$
- Sicherheitskontext-Interpretation $(SecGoal)$

- Ableitung von Sicherheitsmassnahmen ($SecAdaption$)
- Ausführung und Durchsetzung von Sicherheitsmassnahmen ($SecActivity$)

Daraus lassen sich folgende Architekturforderungen ableiten:

- Für mobile Endgeräte mit *Multitasking*-Ausführungsumgebung ergibt sich die Notwendigkeit der Informationssicherheit auf Anwendungsebene, die einheitliche Anbindung von Anwendungen und deren Entkopplung von Kontextverarbeitung.
- Die Multidimensionalität des Sicherheitskontextes erfordert die Aggregation verschiedener Kontextinformationen.
- Die rechenbetonte Sicherheitskontextinterpretation stellt zusätzlich hohe Leistungsanforderungen [CMT07, BDR07].
- Und die Ableitung von Sicherheitsmassnahmen stellt spezifische Flexibiltätsanforderungen.

In Anbetracht der mobilen Nutzungsszenarien bietet sich hierfür der Einsatz einer verteilten Architektur an. In [FK11] schlagen Fischer und Karsch, dass in Abbildung 6.1 dargestellte Architekturmodell vor. Dieses Architekturmodell orientiert sich an bewährte Entwurfsmuster für verteilte Architekturen [CDK03] und kommt den genannten Architekturforderungen wie folgt nach: Die Anbindung von Anwendungen und die Aggregation verschiedener Kontextinformationen wird zentral realisiert (Mobile Device). Die rechenbetonte Sicherheitskontextinterpretation wird in einen dedizierten Kontextdienst, das entsprechende Ressourcen zur Verfügung hat, ausgelagert (Context Server). Die Flexibilitätsanforderungen werden durch die Möglichkeit der Nutzung verschiedener Kontext- und Sicherheitsmodelle adressiert.

Die identifizierten Architekturmerkmale korrespondieren mit allen oben genannten Architekturforderungen. Es können auch andere Entwürfe die Architekturforderungen vollumfänglich adressieren. Der Schwerpunkt liegt allerdings auf der Darlegung der Funktionalität der Einbeziehung von sicherheitsrelevanten Benutzeraktivitäten. Die damit verbundene architektonische Unterstützung soll exemplarisch an diesem vorgestellten Architekturmodell [FK11]

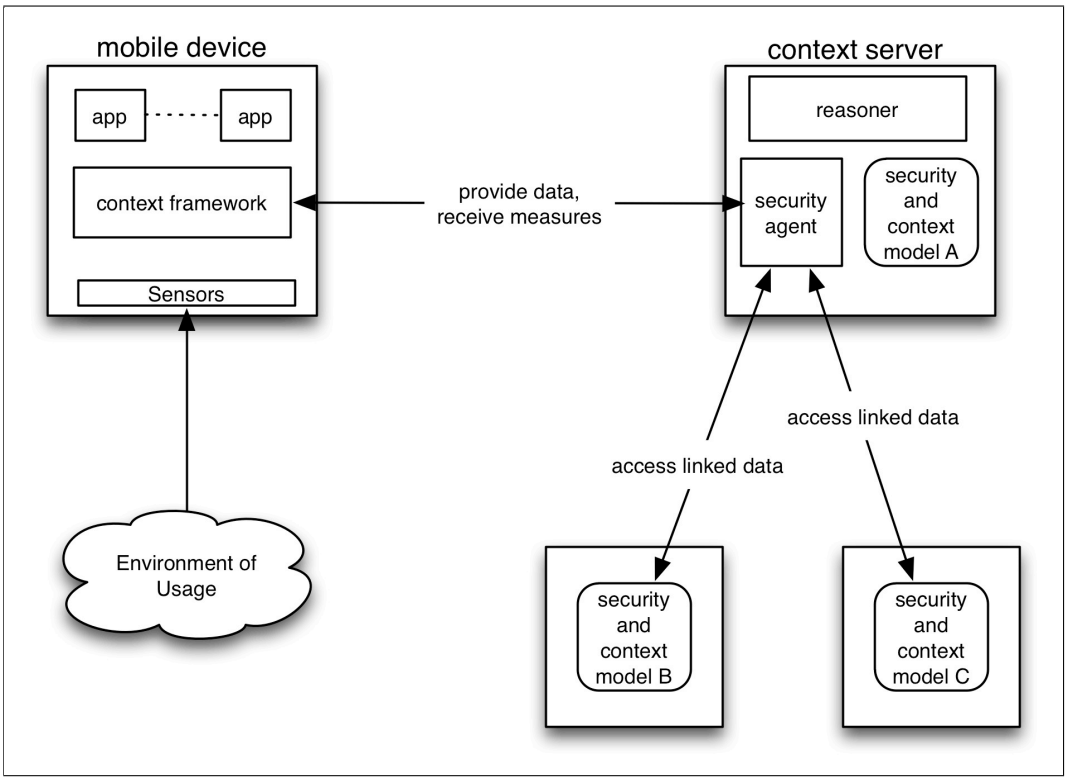


Abb. 6.1.: Architekturmodell nach [FK11]

diskutiert werden. Darauf aufbauend lassen sich folgende Architekturkomponenten mit entsprechenden Verantwortlichkeiten identifizieren:

- Anwendungskomponente (engl. Security Component): In dieser Softwareschicht sind die *lokalen*[1] Anwendungen, die entsprechend der Modellierung (s. Kap. 5.2) die Funktionalität integrieren, Benutzeraktivitäten bezüglich Ihrer Sicherheitsrelevanz (5.6) zu unterscheiden und die Sicherheitsaktivitäten (5.4) zur Verfügung stellen.

- Kontextrahmenwerk (engl. Context Framework): Diese Softwareschicht konsolidiert als *lokale* Middleware die Heterogenität der verschiedenen Kontextquellen und vereinheitlicht die Anwendungsanbindung (s. [CK00, CDK03]). Neben der Ermittlung des Sicherheitskontextes (5.7) stellt sie der Anwendungsebene kontext-spezifische Dienste zur Verfügung.

[1] ff. für die Verortung auf dem mobilen Endgerät

- Kontextserver (engl. Context Service): Der Kontextserver stellt die zentrale Komponente der Systemarchitektur dar (Client/Server-Modell), welche die Verarbeitung und Verwaltung der kontext-spezifischen Informationen *extern* durchführt und darüber hinaus mehrere Akteure in mobilen Nutzungsszenarien simultan bedienen kann.

Eine Referenzimplementierung einer allgemeinen Infrastruktur für Kontextrepräsentation und Interpretation (engl. Context Awareness) insbesondere mit dem Schwerpunkt auf den beiden letztgenannten Architekturkomponenten wird von Müller in [Mül10] vorgestellt. Die Einbeziehung einer Sicherheitsdimension wird von Kobinski in [Kob12] thematisiert. Wobei die Einbeziehung von sicherheitsrelevanten Benutzeraktivitäten keinen detailierten Eingang findet.

Um die wichtigsten Aspekte eines Architekturmodells zu erläutern, das sicherheitsrelevante Benutzeraktivitäten für kontext-adaptives Sicherheitsverhalten heranzieht, soll im Folgenden das Verhalten der Komponenten und deren funktionale Rolle im Zusammenspiel entwickelt werden. Dazu werden die Zustände der einzelnen Komponenten hierarchisch in Form von Verhaltenszustandsdiagrammen (UML) für ein Hauptszenario modelliert ([Kec06, RQ12]). Das betrachtete Szenario stellt eine gewöhnlichen Benutzerinteraktion auf Anwendungsebene dar.

### 6.1.1. Anwendungskomponente

Das dynamische Sicherheitsverhalten von Anwendungen in Reaktion auf die Benutzeraktivität und die Unterscheidung dieser bezüglich Ihrer Sicherheitsrelevanz ist in Abbildung 6.2 dargestellt. Das Kontextrahmenwerk ist der Übersicht halber darin als Einzelzustand[2] abstrahiert dargestellt.

Die eigenständigen Anwendungen führen initial eine *Registrierung* gegenüber dem Kontextrahmenwerk durch. Beide Komponenten registrieren sich gegenseitig als Kommunikationsteilnehmer[3]. Somit wird die Anwendungsan-

[2] engl. Submachine State (UML)
[3] siehe Kapitel 7.1.1 auf Seite 35

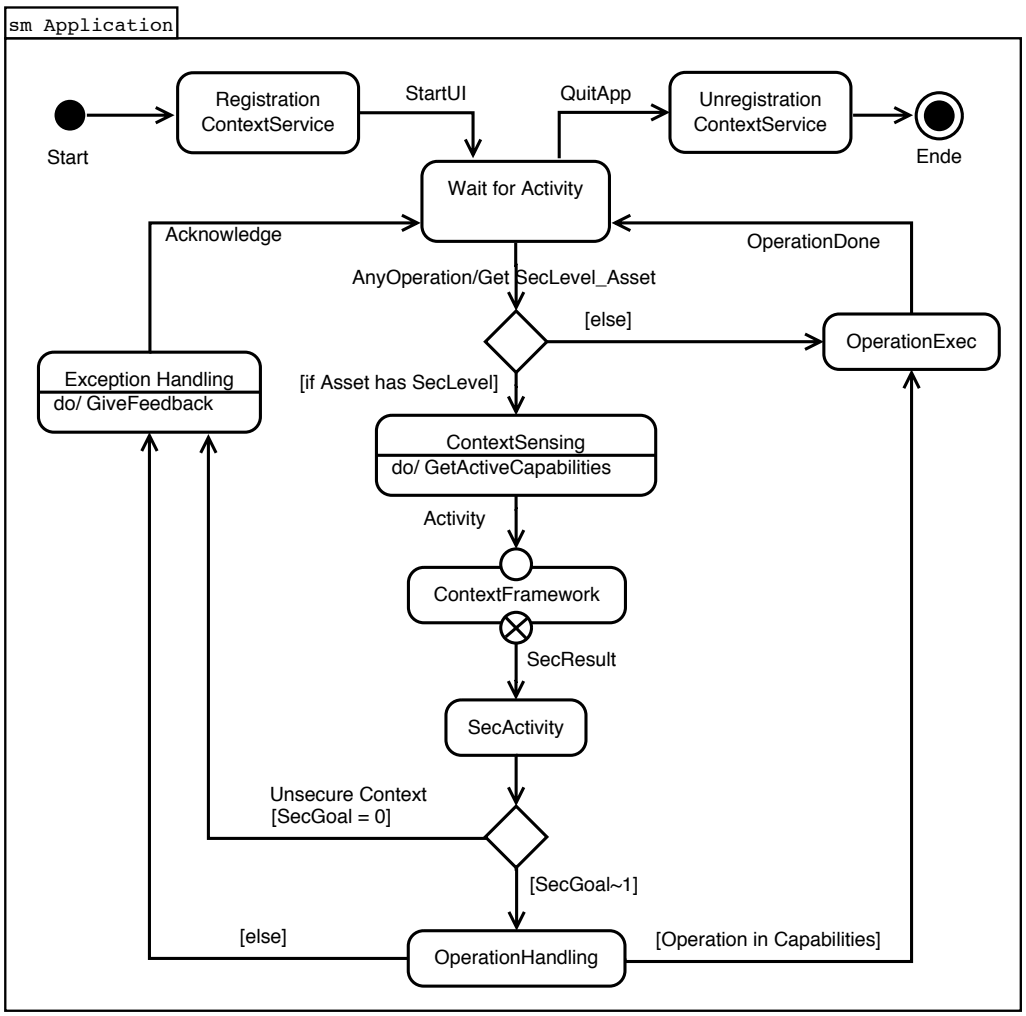


Abb. 6.2.: Verhaltenszustandsdiagramm für die lokale Anwendungsebene (Hauptszenario)

bindung an die Kontextinfrastruktur sichergestellt. Ohne weitere Benutzeraktivitäten wartet eine Anwendung auf eine entsprechend triggernde Interaktion (Wait for Activity). Die ausgelösten Operationen werden zur Unterscheidung mit einer potentiell vorliegenden Sicherheitsklassifizierung der operations-ausgesetzten *Assets* angereichert (Get $SecLevel_{Asset}$). Ist die entsprechende Benutzeraktivität nicht sicherheitsrelevant (*Asset* ohne $SecLevel > 0$), dann kommt die korrespondierende *Operation* unmittelbar zur Ausführung (OperationExec). Die sicherheitsrelevanten *Operationen* werden indes zur weiteren Beurteilung an das lokale Kontextrahmenwerk weiter geleitet, während sie da-

zu mit der aktuellen Sicherheitsmassnahmenkonfiguration ($Capabilities_A$) angereichert werden (GetActiveCapabilities).

Dem lokalen Kontextrahmenwerk stehen insbesondere mit $Capabilities_{App}$, $Capabilities_A$, $Operation$ und $SecLevel$ die obligatorischen Anwendungskontextinformationen zur Verfügung, die die sicherheitsrelevante Aktivitätsdimension repräsentieren (s. (5.6)). Um die Kommunikationskosten niedrig zu halten, wird die vollständige Fähigkeitenliste der Anwendung ($Capabilities_{App}$) dem Kontextrahmenwerk bereits bei der Anwendungsregistrierung[4] zur Verfügung gestellt.

Das lokale Kontextrahmenwerk stellt mit $Capabilities_m \subseteq Capabilities_{App}$ und $SecGoal$ der Anwendung die Informationen zur Verfügung, die es der Anwendung erlaubt die Ausführbarkeit der $Operation$ zu entscheiden (SecResult). Prinzipiell müssen die Möglichkeiten gegeben sein mittels der Sicherheitsadaption respektive der Sicherheitsmassnahmen (s. Kap. 5.2.2), die Sicherheitsanforderung des $Assets$ zu erfüllen. Besteht diese Möglichkeit, so wird mittels Sicherheitsaktivität (5.4) die Sicherheitsadaption mit dem Ziel $Capabilities_m$ durchgeführt (SecActivity). Entspricht $Capabilities_m$ der aktiven Sicherheitsmassnahmenkonfiguration $(Capabilities_A)$, so muss keine Sicherheitsadaption durchgeführt werden. Im Anschluss findet die Entscheidung über die Ausführbarkeit der $Operation$ statt. Steht die $Operation$ nach der Sicherheitsadaption weiterhin zur Verfügung, so kommt diese dann zur Ausführung (OperationExec). Besteht die Möglichkeiten der Sicherheitsanforderungserfüllung nicht $(SecGoal = 0)$, so kommt die entsprechende $Operation$ nicht zur Ausführung (Exception). Um die Bedrohungslage zu entschärfen, werden die ermittelten aber nicht ausreichenden Sicherheitsmassnahmen dennoch angewendet (SecActivity).

### 6.1.2. Kontextrahmenwerk

Das Kontextrahmenwerk wird von der Anwendung hinzugezogen sobald eine sicherheitsrelevante Aktivität detektiert wird. Die Anwendungen sind für das Kontextrahmenwerk reguläre Kontextquellen, die Kontextinformationen zur Verfügung stellen wie andere Kontextquellen auch. Ein wesentlicher Unter-

[4] siehe Kapitel 7.1.1 auf Seite 37 und Anhang A.1.4 auf Seite 95

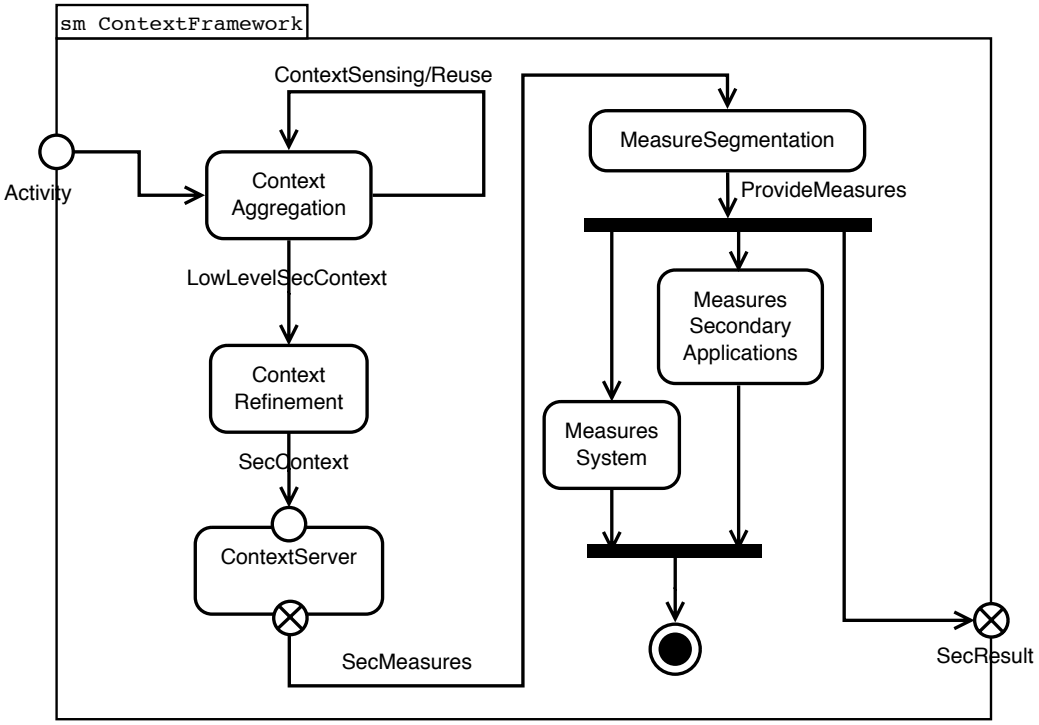


Abb. 6.3.: Verhaltenszustandsdiagramm für das lokale Kontextrahmenwerk (Hauptszenario)

schied ist, dass Anwendungen aktiv Kontextinformationen dem Kontextrahmenwerk bereitstellen und damit ereignis-basiert die Bewertungsaufgabe an das Kontextrahmenwerk weiterleiten. Wie in Abbildung 6.3 dargestellt, wird mit der Aufforderung der Anwendung zur Beurteilung des Kontextes einer Aktivität, dem Kontextrahmenwerk der Anwendungskontext in Form von (5.6) zur Verfügung gestellt. Auf Basis aller angebundenen Kontextquellen wird daraufhin ein aktueller Sicherheitskontext (5.7) ermittelt (Context Aggregation) [FK11]. Mit $Capabilities_{App}$ aller parallel laufenden Anwendungen[5] steht dem Kontextrahmenwerk ebenso der anwendungsübergreifende Ausführungskontext $(AppEnv)$ zur Verfügung. Diese Kontextinformationen fliessen ebenfalls in den aggregierten Sicherheitskontext ein (5.7). Das Kontextrahmenwerk stellt sicher, dass sich diese Aggregation auf *aktuelle* Kontextinformationen stützt, aber dazu nicht bei jeder Anwendungsaufforderung periphere Softwareschichten durchlaufen werden müssen (engl. Context Reuse [VM10]).

[5] *horizontale* Aktivität mittels *Context History* (s. [CLM14, MB04, DAS01, CK00])

Der aggregierte Sicherheitskontext (5.7) wird nach einer Plausibilitätsüberprüfung der Kontextinformationen (Context Refinement), um beispielsweise fehlerhafte Kontextinformationen zu identifizieren, zur weiteren Bewertung an den Kontextserver kommuniziert. Dieser stellt entsprechend als Ergebnis die Sicherheitszielerreichung *SecGoal* und die ermittelten Sicherheitsmassnahmen zur Verfügung (SecMeasures). Die Sicherheitsmassnahmen betreffen nicht nur die anfordernde Anwendung, sondern die vollständige Ausführunggsumgebung. Neben den anwendungs-spezifischen Sicherheitsmassnahmen $Capabilities_m$ werden auch weitere Sicherheitsmassnahmen $(GlobalCapabilities)$ kommuniziert. Wird beispielsweise die gleichzeitige Nutzung bestimmter Anwendungen als Konflikt bewertet, so werden entsprechende Sicherheitsmassnahmen an die *sekundären* Anwendungen kommuniziert. Ebenso sind Sicherheitsmassnahmen auf der Systemebene verortet, um anwendungs-übergreifende Funktionalitäten[6] entsprechend zu steuern. Abschliessend werden die anwendungs-spezifischen Sicherheitsmassnahmen $Capabilities_m$ und die Bewertung der Sicherheitszielerreichung *SecGoal* der anfordernden Anwendung als Bewertungsantwort zur Verfügung gestellt (SecResult).

Das lokale Kontextrahmenwerk fungiert als verbindende Komponente mit einheitlich spezifizierten Schnittstellen zwischen der Anwendungskomponente und dem Kontextserver und bindet diverse *lokale* Kontextquellen und das Betriebssystem in die Kommunikation mit ein (s. [FK11, Mül10, CDK03]).

Neben den hier skizzierten Hauptszenario, ist eine weitere Kernaufgabe des Kontextrahmenwerks, die des *Monitorings* des Sicherheitskontextes (s. [VM10, EPS10]). Sicherheitskontextänderungen können von allen Kontextquellen herrühren. Entscheidend sind dabei nur die veränderten Interpretationsergebnisse [BBH+09]. Daher werden die Sicherheitskontextänderungen *fortlaufend* zur Kontextinterpretation dem Kontextserver zur Verfügung gestellt. Entsprechend ermittelte Sicherheitsmassnahmenänderungen werden aktiv an die System- und der Anwendungsebene kommuniziert (s. [CK00]). Dieser interne Prozess stellt sicher, dass eine persistente Kontextwahrnehmung respektive ein Kontextbewusstsein (engl. Context Awareness) gewährleistet wird (s. [ADB+99]).

[6] beispielsweise die Durchführung von Bildschirmaufnahmen

### 6.1.3. Kontextserver

Der Kontextdienst auf dem externen Kontextserver wird von dem lokalen Kontextrahmenwerk hinzugezogen, sobald eine Bewertung eines Sicherheitskontextes vorgenommen werden soll. Wie in Abbildung 6.4 dargestellt, werden für die Bewertung die entsprechend genutzten Kontext- und Sicherheitsmodelle zentral auf dem Kontextserver angewendet. Insbesondere wenn neue Zusammenhänge zwischen den Kontextinformationen eingearbeitet werden, stehen diese Aktualisierungen unmittelbar jedem mobilen Endgerät zur Verfügung. Der Kontextserver speichert zusätzlich die kommunizierten Sicherheitskontexte respektive die vorangegangenen Auswertungen[7] aller Anwendungen ab (s. [CLM14, MB04, DAS01, CK00]). Die chronologischen Kontextinformationen ermöglichen die Implementierung und Anwendung dynamischer Kontext- und Sicherheitsmodelle (engl. Learning Algorithms [BDR07]) in dem Beziehun-

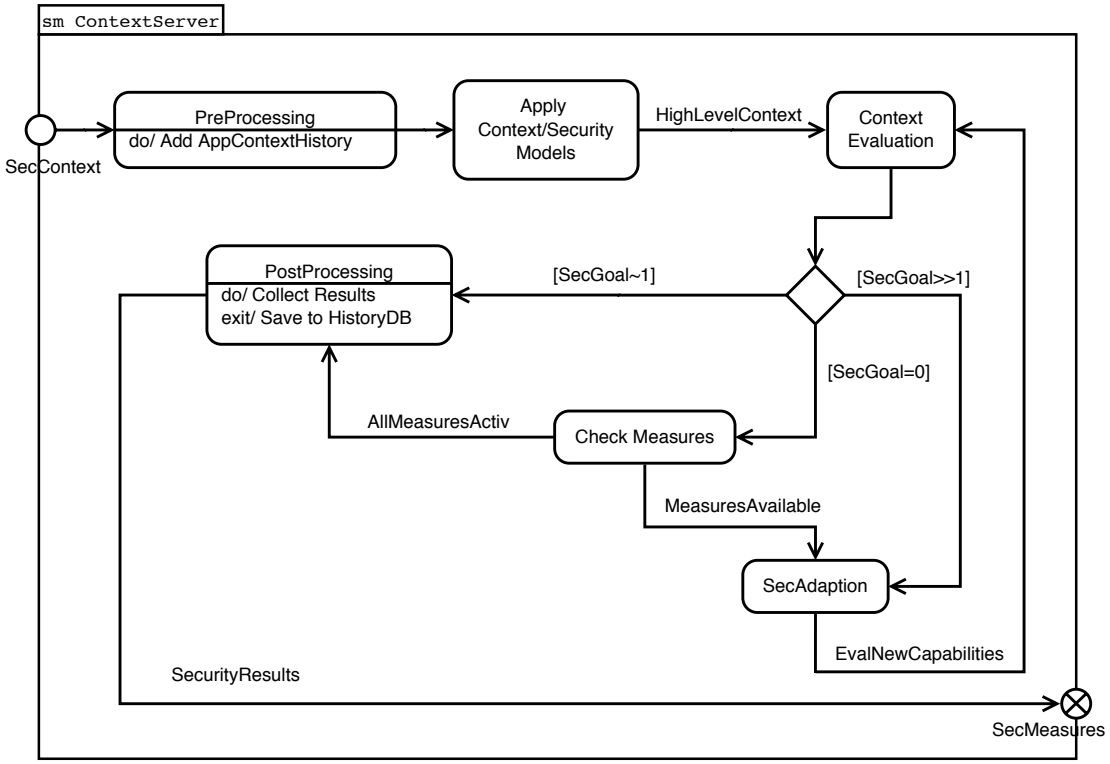


Abb. 6.4.: Verhaltenszustandsdiagramm für den externen Kontextserver (Hauptszenario)

[7] engl. Context History

gen zwischen Kontextinformationen identifiziert werden [VM10]. Antizipierte Bedrohungslagen sind somit im Bewertungsprozess aktueller Sicherheitskontexte nutzbar und proaktive Sicherheitsmassnahmen werden dadurch kommunizierbar (s. [CCB08]).

Der Bewertungsvorgang wird von einem Service-*Agenten* auf dem externen Kontextserver orchestriert [Kob12, FK11, Mül10]. Es werden die Historien-Informationen hinzugezogen (PreProcessing), die ausgewählten Kontext- und Sicherheitsmodelle angewendet und auf Basis dieser *High-Level*-Ergebnisse (s. [YDM12, BBH+09, DAS01]) wird der kommunizierte aktive Sicherheitskontext hinsichtlich potenzieller Bedrohungen bewertet (Context Evaluation). Eine Benutzeraktivtität wird somit anhand der spezifischen Modelle[8] in bekannten Bedrohungslagen identifizierbar gemacht. Kann das inhärente Schutzziel der Benutzeraktivtität nicht in dem kommunizierten Sicherheitskontext erreicht werden $(SecGoal = 0)$, so wird eine Sicherheitsadaption in Form von $Capabilities_m$ ermittelt (SecAdaption), die eine Sicherheitskontextänderung im Sinne des Schutzziels vornimmt (s. Kap. 5.2.4). Dies stellt eine vorweggenommene Bewertung dar, da die Sicherheitsadaption noch nicht abschliessend auf dem mobilen Endgerät umgesetzt wurde. Die Bewertung der Schutzzielerreichung und die Sicherheitsmassnahmen $(Capabilities_m)$ werden auf dem Kontextserver in einer Historien-Datenbank gesichert (PostProcessing) und als Antwort dem Kontextrahmenwerk des mobilen Endgeräts zur Verfügung gestellt (SecMeasures). Deregistriert sich eine Anwendung auf dem mobilen Endgerät beim Kontextrahmenwerk – weil die Anwendung beendet wird – dann stellt das Kontextrahmenwerk sicher, dass der externe Kontextserver entsprechend notifiziert wird und die korrespondierenden Historien-Einträge entfernt werden.

## 6.2. Zusammenfassung

In den vorangegangenen Abschnitten wird eine Architektur vorgestellt, die sicherheitsrelevante Benutzeraktivitäten für kontext-adaptives Sicherheitsverhalten integriert. Dazu werden die essentiellen Merkmale modelliert, welche

[8] Öffentliche-, unternehmens-, projekt- oder personen-spezifische Sicherheits- und Kontextmodelle (nach [FK11])

die notwendige architektonische Unterstützung repräsentieren. Es wird das Zusammenspiel der einzelnen Architekturkomponenten dargelegt und deren Verantwortlichkeiten anhand von Verhaltensmodellen entwickelt. Die funktionale Rolle der einzelnen Komponenten wird bewusst abstrakt vorgestellt, um sich auf die wesentlichen Einheiten zu konzentrieren, und somit zu einem besseren Verständnis der Integration der Kontextquelle *Benutzeraktivität* beizutragen. Die Erweiterbarkeit wird durch die klare Separierung der Verantwortlichkeiten der entsprechenden Architekturkomponenten gewährleistet.

Aufgrund der Schutzzielvorgaben korreliert das kontext-adaptive Sicherheitsverhalten eng mit den sicherheitsrelevanten Benutzeraktivitäten. Entsprechend wird der Sicherheitsadaptionsprozess gleichermassen dargelegt.

Wie aus der vorangehenden Vorstellung ersichtlich, wird die Einbeziehung von sicherheitsrelevanter Benutzeraktivität nur durch die vollständige Unterstützung der Infrastruktur inklusive der Anwendungen ermöglicht. Im nächsten Kapitel wird die hier vorgestellte Anwendungskomponente (Security Component) mit ihren spezifischen Eigenschaften detailierter vorgestellt.

# 7. Anwendungsintegration

Da sich allgemeine oder individuelle Software-Anwendungen per se von einander Unterscheiden, stellen die spezifischen Angriffsflächen (s. Kap. 5.2.1) in ihrer Summe eine unüberschaubare Menge dar. Entsprechend gilt dies auch für das gewünschte kontext-adaptive Sicherheitsverhalten von Anwendungen. Ein anwendungs-spezifisches Kontext- und Adaptionsmodell kann daher nur generalisiert vorgestellt werden. Die tatsächliche Integration von kontext-adaptivem Sicherheitsverhalten und der Kommunikation des Anwendungskontextes, die beide von der Anwendungskomponente (Security Component) gekapselt werden (s. Kap. 6.1.1), bleibt in der Verantwortung der Anwendungsentwicklung. Die Infrastrukturunterstützung erfolgt über eine wohldefinierte Schnittstelle (API). Im Folgenden soll ein generisches Kontext- und Adaptionsmodell, und die Integration von Anwendungen mit kontex-sensitivem Sicherheitsverhalten vorgestellt werden.

## 7.1. Interaktion

### 7.1.1. Interaktionsmodell

Neben der adaptiven Sicherheitsfunktionalität und der Ermittlung von sicherheitsrelevanter Benutzeraktivitäten, muss die Anwendungskomponente (engl. Security Component) der einzelnen Anwendungen zur Anbindung an die vorgestellte Infrastruktur eine Schnittstelle anbieten und sich an die Schnittstelle der Kontextinfrastruktur anbinden lassen. Exemplarisch sollen vorhandene Standards zum Einsatz herangezogen werden. Insbesondere wird HTTP

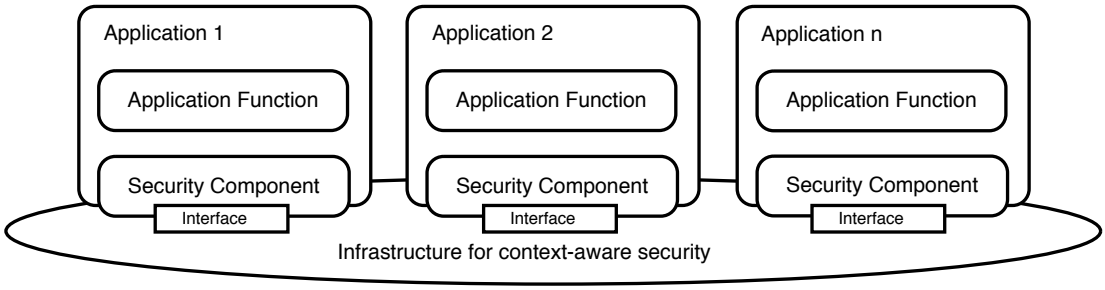


Abb. 7.1.: Die Anwendungskomponente (Security Component) implementiert die Integrationfunktionalität und kapselt die Logik dieser Kontextinformationsquelle.

[FGM+99] für den Transport und XML [BPSM+06] für die Kodierung der Kommunikation genutzt. Für die Interprozesskommunikation (IPC) zwischen der Anwendungskomponente und dem Kontextrahmenwerk bietet sich das leichtgewichtige XML-RPC [Win03] besonders an. Neben der Adressierung einer schlanken und performativen Kommunikation bietet es eine leichtgewichtige Möglichkeit der Integration in Anwendungen an.

Das in Form eines W3C XML-*Schema* [W3C12] entwickelte und spezifizierte Datenmodell besteht aus dem XML-RPC Datenmodell und integriert zusätzlich anwendungskontext-spezifische Erweiterungen[1]. Dieses von der Anwendungskomponente (Security Component) und dem Kontextrahmenwerk verwendete XML-*Schema* ist als Quelltext des Umfangs wegen im Anhang A.1.1 ab Seite 83 vollständig aufgelistet.

Die Anwendungskomponente und das Kontextrahmenwerk sind wie in Kapitel 6 vorgestellt auf dem lokalen Endgerät verortet. Somit wäre auch eine *lokale* Interprozesskommunikationsmethode einsetzbar. Diese würde die architektonische Flexibilität, die XML-RPC bietet, allerdings nicht erhalten. Denn ein Kontextrahmenwerk muss nicht zwingend *lokal* zum Einsatz kommen[2].

[1] Die Erweiterungen werden im Kapitel 7.3 ab Seite 46 vorgestellt.

[2] Eine erweiterbare Architektur adressiert die aktuellen Entwicklungen in dem Bereich des *Internet der Dinge* (IoT), der Gebäudeautomation und des *Wearable Computings*, die bereits erweiterte mobile Interaktionsszenarien zur Verfügung stellen.

### Schnittstelle

Für die Anbindung soll das genutzte Interaktionsmodell dem nachrichten-basierten *Publish-Subscribe*-Paradigma folgen. Für die Integration müssen die folgenden abstrakten Schnittstellen-Funktionen von jeder Anwendung implementiert werden (s. Abb. 7.1):

- $PublicAppChannelforSecActivity()$
- $PublicAppUnsubscribeActivity()$
- $SecResultNotifyCb()$

Sowie die Gegenstelle — das Kontextrahmenwerk — muss folgende abstrakte Schnittstellen-Funktionen für die Anwendungsanbindung implementieren:

- $PublicServiceChannelforMeasures()$
- $PublicServiceUnsubscribeMeasures()$
- $ActivityNotifyCb()$

Um anwendungsseitig Redundanzen zu vermeiden empfiehlt sich der Einsatz einer dynamischen Bibliothek.

Die Schnittstelle definiert zum einen Aufrufe, die jede Anwendung bereitstellen muss und zum anderen Aufrufe die vom Kontextrahmenwerk zusätzlich bereitgestellt werden. Diese werden von der Anwendung aufgerufen und vice versa. Dadurch wird eine bidirektionale Kommunikation ermöglicht. Die $App/ServiceChannel$- und $App/ServiceUnsubscribe$-Aufrufe werden für die gegenseitige De-/Registrierung verwendet (s. Kap. 6.1.1). Somit wird die Anwendung als Quelle für Anwendungskontextinformationen und das Kontextrahmenwerk als Quelle für anwendungsspezifische Sicherheitsmassnahmen aktiviert (s. Abb. 7.2)[3] respektive deaktiviert. Der Aufruf $ActivityNotifyCb()$ stellt eine vom Kontextrahmenwerk bereitgestellte Callback-Funktion dar, die von

[3] siehe auch den Registrierungsaufruf im Anhang A.1.4 auf Seite 95.

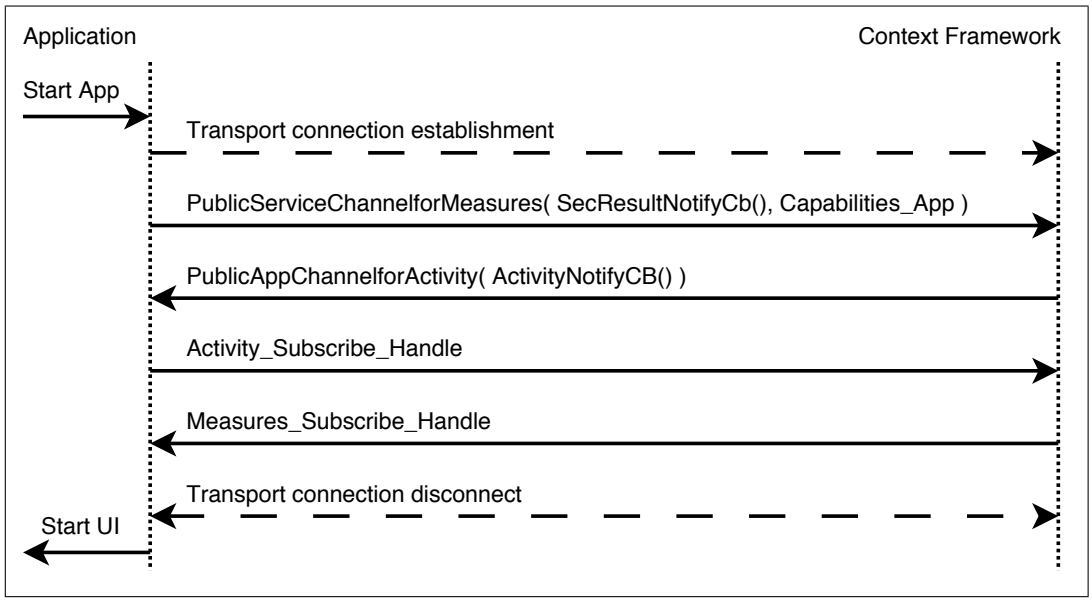


Abb. 7.2.: Registrierungsablauf (Registration ContextService)

der Anwendung für die Notifizierung des Kontextrahmenwerks über Benutzeraktivitäten genutzt wird (Activity). Der *SecResultNotifyCb*()-Aufruf stellt eine von der Anwendung für das Kontextrahmenwerk bereitgestellte Callback-Funktion dar, um kontext-sensitiv die Anwendung über Sicherheitsmassnahmen zu benachrichtigen (SecResult).

Aufgrund der ereignis-basierten Kommunikation stellt die Anwendung keine zusätzliche Möglichkeit einer Aktivitätsabfrage zur Verfügung. Benutzeraktivitäten werden aktiv von der Anwendung kommuniziert und stehen somit unmittelbar zur Verfügung. Wie das dazugehörige Kommunikationsmodell aussieht, wird im Folgenden vorgestellt.

### 7.1.2. Kommunikationsmodell

#### Benutzeraktivität

Das Kontextrahmenwerk als zentrale Komponente auf dem mobilen Endgerät ist von allen Anwendungen lokal mittels TCP-*Socket* [RFC81] und einem

*Endpunkt* mit *privilegierten* Rechten ($Port < 1024$) [IAN16] erreichbar (FRAMEWORK_STATIC_LOWPORT). Die Notifizierung des Kontextrahmenwerks mittels XML-RPC bezüglich einer sicherheitsrelevanten Benutzeraktivität via $ActivityNotifyCb()$-Aufruf sieht dazu strukturell beispielsweise wie folgt aus:

```
1  POST /service/appcontext HTTP/1.1
2  User-Agent: APP_NAME/APP_VERSION
3  Host: localhost:FRAMEWORK_STATIC_LOWPORT
4  Content-Type: text/xml
5  Content-length: 732
6
7  <?xml version="1.0" encoding="UTF-8"?>
8  <methodCall>
9    <methodName>APP_REG_ID.ActivityNotifyCb</methodName>
10 <!-- START passing values -->
11    <params>
12
13 <!-- START First argument for metadata of the notify -->
14     <param>
15       <value>
16         <struct>
17         <!-- PLACEHOLDER -->
18         </struct>
19       </value>
20     </param>
21 <!-- END First argument for metadata of the notify -->
22
23
24 <!-- START 2nd argument for application context (or capabilities informations) -->
25     <param>
26       <value>
27         <struct>
28         <!-- PLACEHOLDER -->
29         </struct>
30       </value>
31     </param>
32 <!-- END 2nd argument for application context (or capabilities informations) -->
33
34    </params>
35 <!-- END passing values -->
36 </methodCall>
```

Quelltext 7.1: Benachrichtigungsstruktur für eine Benutzeraktivität

Die zugehörige URI sieht wie folgt aus:

```
http://localhost:FRAMEWORK_STATIC_LOWPORT/service/appcontext
```

Wie im Quelltext 7.1 dargelegt (XML-RPC), werden genau zwei Argumente übergeben. Der Datentyp der Argumente ist ein Verbund (*struct*) von mehreren unterschiedlichen Komponenten mit ungleichen Datentypen. Der erste Verbund (erstes Argument) beinhaltet Metadaten zur kommunizierten Benutzeraktivität. Dies sind beispielsweise eindeutige Bezeichner der Anwendung und der kommunizierten Benutzeraktivität sowie ein Zeitstempel. Für die Benutzeraktivitätsbenachrichtigung werden dazu folgende Schlüsselnamen verwendet:

- ACTIVITY_APPLICATION_ID
- ACTIVITY_UUID
- ACTIVITY_APP_TIMESTAMP

Der zweite Verbund (zweites Argument) beinhaltet die Anwendungskontextinformationen der Benutzeraktivität[4].

Neben der Benutzeraktivität respektive dem Anwendungskontext muss die Anwendung auch die zur Verfügung stehenden anwendungs-spezifischen Sicherheitsmassnahmen kommunizieren. Die zur Verfügung stehenden anwendungs-spezifischen Sicherheitsmassnahmen $Capabilities_{App}$ werden in der Regel während der Kommunikationsregistrierung einmalig kommuniziert (s. Kap. 7.1.1). Davon zu unterscheiden ist die aktuelle Sicherheitsmassnahmenkonfiguration[5] $Capabilities_A$ in der Anwendung. Diese wird als Anwendungskontextinformation (zweites Argument) zusammen mit der Benutzeraktivität stets mit kommuniziert (s. (5.8) im Kap. 5.2.4).

Die empfangenen Anwendungskontextinformationen werden stets bestätigt. Die Übermittlung der Benutzeraktivität wird vom Kontextrahmenwerk bei Erfolg positiv wie folgt quittiert (XML-RPC):

```
1 HTTP/1.1 200 OK
2 Connection: close
3 Content-Length: 223
```

[4] siehe Kapitel 7.3.1 ab Seite 46
[5] siehe Kapitel 7.3.2 ab Seite 54

```
4  Content-Type: text/xml
5  Date: Fri, 18 Mar 2016 16:47:30 UTC
6  Server: CONTEXT_FRAMEWORK/FRAMEWORK_VERSION
7
8  <?xml version="1.0" encoding="UTF-8"?>
9  <methodResponse>
10     <params>
11         <param>
12             <value>
13                 <string>OK</string>
14                 </value>
15             </param>
16         </params>
17 </methodResponse>
```

Quelltext 7.2: Empfangsbestätigung der Benutzeraktivität

Ein validierter vollständiger XML-RPC-Aufruf für die Benachrichtigung des Kontextrahmenswerks bezüglich einer sicherheitsrelevanten Benutzeraktivität befindet sich des Umfangs wegen im Anhang A.1.2.

### Sicherheitsmassnahmen

Die Anwendungen stellen bei ihrer Ausführung einen willkürlichen, freien TCP-*Endpunkt* (Port) für die Kommunikation mit dem Kontextrahmenwerk zur Verfügung. Somit wird von jeder einzelnen Anwendung ein dedizierter Endpunkt zur Verfügung gestellt. Da Anwendungen in der Regel mit nicht-privilegierten Rechten ausgeführt werden, sind diese Endpunkte im Bereich ab 1024 aufwärts angesiedelt [IAN16]. Das Kontextrahmenwerk bekommt im Rahmen der Kommunikationsregistrierung via der Callback-Lokation diesen Endpunkt (*APP_RANDOM_HIGHPORT*) mitgeteilt. Die Notifizierung einer Anwendung mittels XML-RPC bezüglich einer Sicherheitsmassnahme via $SecResultNotifyCb()$-Aufruf sieht strukturell beispielsweise wie folgt aus:

```
1  POST /service/appmeasures HTTP/1.1
2  User-Agent: CONTEXT_FRAMEWORK/FRAMEWORK_VERSION
3  Host: localhost:APP_RANDOM_HIGHPORT
4  Content-Type: text/xml
5  Content-length: 737
6
7  <?xml version="1.0" encoding="UTF-8"?>
```

```
<!-- Response Callback for Measures -->
<methodCall>
  <methodName>FRAMEWORK_REG_ID.SecResultNotifyCb</methodName>
<!-- START passing values -->
   <params>

<!-- START First argument for metadata of the security response -->
    <param>
      <value>
        <struct>
        <!-- PLACEHOLDER -->
        </struct>
      </value>
    </param>
<!-- END First argument for metadata of the security response -->


<!-- START 2nd argument for application measures -->
    <param>
      <value>
        <struct>
        <!-- PLACEHOLDER -->
        </struct>
      </value>
    </param>
<!-- END 2nd argument for application measures -->

   </params>
<!-- END passing values -->
</methodCall>
```

Quelltext 7.3: Benachrichtigungsstruktur für Sicherheitsmassnahmen

Die zugehörige URI sieht wie folgt aus:

```
http://localhost:APP_RANDOM_HIGHPORT/service/appmeasures
```

Für die Kommunikation der anzuwendenden Sicherheitsmassnahmen werden, wie im Quelltext 7.3 dargelegt (XML-RPC), ebenso genau zwei Argumente übergeben. Der Datentyp der Argumente ist ein Verbund (*struct*) von mehreren unterschiedlichen Komponenten mit ungleichen Datentypen. Der erste Verbund (erstes Argument) beinhaltet Metadaten zu den Sicherheitsmassnahmen. Dies sind vor allem der eindeutige Bezeichner der korrespondierenden Benutzeraktivität als Referenz (ACTIVITY_UUID), die Priorisierung der Sicherheitsmassnahmen und die Sicherheitszielerreichung (s. (5.11) auf S. 21). Für die

Sicherheitsmassnahmenbenachrichtigung werden dazu folgende Schlüsselnamen verwendet:

- MEASURES_REF_ACTIVITY_UUID
- MEASURES_PRIORITY
- MEASURES_SECGOAL

Der zweite Verbund (zweites Argument) beinhaltet die Informationen bezüglich der anwendungsspezifischen Sicherheitsmassnahmen[6].

Die empfangenen Sicherheitsmassnahmen werden stets bestätigt. Die Übermittlung von Sicherheitsmassnahmen wird von der Anwendung bei Erfolg positiv quittiert (XML-RPC):

```
1  HTTP/1.1 200 OK
2  Connection: close
3  Content-Length: 223
4  Content-Type: text/xml
5  Date: Fri, 18 Mar 2016 16:47:31 UTC
6  Server: APP_NAME/APP_VERSION
7
8  <?xml version="1.0" encoding="UTF-8"?>
9  <methodResponse>
10     <params>
11         <param>
12             <value>
13                 <string>OK</string>
14                 </value>
15             </param>
16         </params>
17 </methodResponse>
```

Quelltext 7.4: Empfangsbestätigung der Sicherheitsmassnahmen

Ein validierter vollständiger XML-RPC-Aufruf für die Benachrichtigung der Anwendungskomponente bezüglich spezifischer Sicherheitsmassnahmen befindet sich des Umfangs wegen im Anhang A.1.3.

[6] siehe Kapitel 7.3.2 auf Seite 54

Sowohl die Kommunikation der Benutzeraktivitäten als auch die der Sicherheitsmassnahmen nutzen Datenverbünde. Die Nutzung von Datenverbünden hat den Vorteil, dass ein *Schlüssel* die einzelnen Datensätze identifiziert und somit Erweiterungen integriert werden können, ohne dass diese mit einer rudimentären Unterstützung der Kommunikation – wie sie hier vorgestellt wird – interferieren (s. [Win03]). Dadurch können zusätzliche Anforderungen einer spezifischen Infrastruktur für kontext-sensitive Sicherheit ohne Weiteres adressiert werden.

In den vorangegangenen Abschnitten lag der Fokus auf der Integration von Anwendungen in die vorgestellte Infrastruktur für kontext-sensitive Sicherheit (s. Kap. 6). In den nächsten Abschnitten wird die Integration der Anwendungskomponente (Security Component) in die einzelne Anwendung – wie sie im Kapitel 6.1.1 beschrieben ist – vorgestellt. Zur Verortung der Anwendungskomponente werden zunächst die strukturellen Rahmenbedingungen beleuchtet.

## 7.2. Anwendungsarchitektur

Um in einer Benutzeranwendung die Aktivitäten und deren Sicherheitsrelevanz zugänglich zu machen, muss die innere Anwendungsgliederung mit ihren Funktionalitäten identifiziert werden. Eine allgemein akzeptierte Struktur für grafische Benutzeranwendungen ist das sogenannte *Model-View-Controller* Entwurfsmuster (MVC). Dieses Entwurfsmuster sieht zur Komplexitätsreduzierung von Anwendungen eine Dreiteilung der Funktionalitäten vor. Die jeweils verschiedenen Zuständigkeiten finden sich in einer Modell- (Model), einer Präsentations- (View) und in einer Steuerungskomponente (Controller) wieder [Ree79]. Die Modellkomponente repräsentiert die Assets und die vollständige Anwendungslogik im Sinne der Aufgabenstellung, die die Anwendung adressiert. Die Steuerungskomponente implementiert zentral die *Logik* der Benutzerschnittstelle. Die Präsentationskomponente *visualisiert* in der grafischen Benutzeroberfläche den gegenwärtigen Zustand der Modellkomponente.

Die Steuerungskomponente nimmt dazu die Benutzeraktivitäten auf der gra-

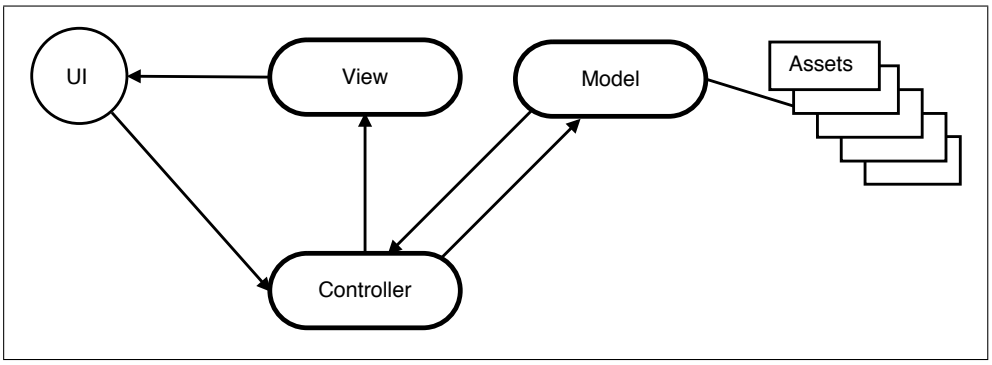


Abb. 7.3.: Schematische Darstellung des MVC-Entwurfsmusters für GUI's

fischen Benutzeroberfläche (und anderer HID's) entgegen und leitet diese *Ereignisse* an die Modellkomponente weiter. Entsprechend der Ereignisse führt die Anwendung korrespondierende Aktionen durch. Daraufhin weist die Steuerungskomponente die Präsentationskomponente an, eine Aktualisierung der Benutzerschnittstelle durchzuführen, um den veränderte Zustand der Modellkomponente respektive der Anwendung zu reflektieren [GHJV94].

Die Modellkomponente (Model) stellt der Steuerungskomponente (Controller) eine einheitliche Schnittstelle für den Zugriff auf die Anwendungslogik und die Assets zur Verfügung[7]. Bezüglich der Benutzeraktivität steht diese ebenso im direkten Dialog mit der Steuerungskomponente. Wie in Abbildung 7.3 schematisch dargestellt, ist diese Interaktion ganz unabhängig von der dahinter liegenden Asset-Datenstruktur. Mit Blick auf die Informationssicherheit sind die Assets allerdings von besonderem Interesse. Ebenso entkoppelt ist die Präsentationskomponente (View), die für die Rückmeldung an den Benutzer verantwortlich ist. Die Präsentation der Assets ist vollständig unabhängig von der Datenrepräsentation der Modellkomponente respektive von der dahinter liegenden Asset-Datenstruktur. Faktisch *interpretiert* die Steuerungskomponente (Controller) für die Präsentationskomponente (View) den gegenwärtigen Zustand der Modellkomponente (Model) und vice versa *interpretiert* die Steuerungskomponente die Benutzeraktivität für die Modellkomponente (Model).

Im nächsten Abschnitt soll ein generisches Anwendungskontext- und Adap-

[7] MVC ausprägungsabhängig (s. [iOS15, BMR+96, KP88])

tionsmodell vorgestellt und in diesem vorgestellten Entwurfsmuster für Anwendungen mit grafischer Benutzerschnittstelle verortet werden.

## 7.3. Kontext-/Adaptionsmodell

Als essenzieller Anteil des Sicherheitskontextes kommunizieren die sicherheitsrelevanten Benutzeraktivitäten wichtige Kontextattribute (s. Kap. 5.2.4). Die adaptive Sicherheitsfunktionalität hat wiederum Einfluss auf das Anwendungsverhalten. Für die Kontextinformationsverarbeitung müssen diese Anwendungskontextinformationen explizit formal zugänglich gemacht und die Sicherheitsfunktionalität repräsentiert werden. Für die Veranschaulichung der Datenmodellierung soll der Zweckmäßigkeit halber die erweiterbare Auszeichnungssprache XML [BPSM+06] herangezogen werden. Dazu werden in Form eines W3C XML-*Schema* [W3C12] die formale Struktur der Modellbestandteile spezifiziert und in den nachfolgenden Kapiteln beispielhaft eingesetzt. Im nächsten Abschnitt wird ein generisches Anwendungskontextmodell für die Kontextquelle *Benutzeraktivitäten* vorgestellt.

### 7.3.1. Anwendungskontextmodell

Die Operationen und Kommunikationskanäle stellen neben den Assets, wie in Kapitel 5.2.1 formuliert, die wichtigsten Kontextattribute (Interaktionsmerkmale) des Anwendungskontextes dar, die die Sicherheitsimplikationen bestimmen. Software-technisch betrachtet stellen die Operationen keine direkte Abbildung auf beispielsweise *Methoden* dar. Vielmehr sind es abstrahierte Kapselungen die für einen Benutzer eine Anwendungssemantik repräsentieren. Diese korrelieren mit der Nutzungsmöglichkeit, welche die grafische Benutzerschnittstelle zur Verfügung stellt. Für eine Identifikation der relevanten Kapselungen empfiehlt sich methodisch eine deduktive Herangehensweise.

### Operationen

Das Ereignismodell für grafische Benutzerschnittstellen (s. Kap. 7.2) weist einen hohen Detaillierungsgrad auf, was wiederum für die Modellierung von durchzuführenden Operationen keine adäquate Abbildung erlaubt. Auch eine *Ereignisfilterung* in der Steuerungskomponente (MVC) ist nicht ausreichend. Ein verminderter Detaillierungsgrad adressiert die Abstraktionsforderung effektiver. Dazu werden in Anlehnung an CRUD-Operationen [Mar83] die folgenden Merkmale zur Modellierung von *Operationen* vorgestellt, die auf einer höheren Abstraktionsebene relevante und klassifizierte Kontextinformationen bereitstellen:

- *Attribute* - Metadaten Änderungen der Assets
- *Close* - Zugriffsende auf Assets
- *Create* - Erstellung von Assets
- *Delete* - Löschung von Assets
- *Move* - Lokationsänderung der Assets
- *Modify* - Inhaltsänderung der Assets
- *Open* - Zugriff auf Assets
- *Read* - Lesezugriff auf Assets
- *Write* - Schreibzugriff auf Assets

Da die benutzerschnittstellen-spezifischen Ereignisinformation in der Steuerungskomponente ihre Destination finden, sind die oben genannten Kontextattributmerkmale insbesonders in der Modellkomponente (Model) der Anwendung identifizierbar. Die Anwendungsentwicklung identifiziert und klassifiziert demnach die relevanten *Aufrufe*. Besonders strukturierte Assets können auch partiell den oben genanten Operationen unterworfen sein. Dementsprechend wird eine feingranulare Ermittlung der Benutzeraktivitäten ermöglicht.

```
1  <xs:complexType name="ValueType" mixed="true">
2   <xs:choice>
3    ...
4    <xs:element name="operation" type="ContextActivityOperation" />
5    ...
6   </xs:choice>
7  </xs:complexType>
8
9  <xs:simpleType name="ContextActivityOperation">
10  <xs:restriction base="ASCIIString">
11   <xs:pattern value="attribute|close|create|delete|move|modify|open|read|write" />
12  </xs:restriction>
13 </xs:simpleType>
```

Quelltext 7.5: Datentyp XML-*Schema* Definition für *Operation*

Zur Repräsentation des Kontextattributes *Operation* gelten die vorgestellten Merkmale als Wertemenge. Für die formale Beschreibung dieses Modellbestandteils kommt ein XML-*Schema* zum Einsatz. Für die Modellierung wird wie im Quelltext 7.5 dargestellt ein neuer Datentyp definiert, der das Datenmodell aus Kapitel 7.1.1 erweitert[8]. Für die Benutzeraktivitätsbenachrichtigung wird folgender Schlüsselname verwendet:

- ACTIVITY_OPERATION

Die Beziehungen zwischen den Werten und deren sicherheits-implikativen Gewichtungen werden explizit *nicht* in der Anwendungskomponente modelliert. Geeigneter ist dafür das Kontextrahmenwerk, da es dies zentral und einheitlich für alle Anwendungen vornehmen kann. Solch eine Sicherheitskonfiguration kann auch vom Kontextserver übernommen werden, so dass diese geräte-übergreifend angewendet werden kann (Security Policy).

### Assets

Der Schutzbedarf von Assets wird als Sicherheitsklassifizierung modelliert und trägt als Kontextattribut zum Sicherheitskontext bei. Die Sicherheitsklassifizie-

[8] Das vollständige erweiterte XML-Schema Datenmodell ist im Anhang A.1.1 auf Seite 83 aufgelistet.

rung der Assets (*SecLevelClass*) erfolgt über die Anreicherung der Assets mittels Sicherheitsmetadaten. Dies erfordert, dass die daten-repräsentierende Struktur der Modellkomponente erweitert und die persistente Speicherung dieser Sicherheitsmetadaten gewährleistet wird. Die Modellkomponente (MVC) erlaubt dies zentral vorzunehmen und abstrahiert die unterschiedlichen asset-spezifischen Strukturen (Interpretationlogik). Die Sicherheitsmetadaten sollten dabei verteilt und asset-gebunden gesichert werden, um den Erhalt dieser Sicherheitsmetadaten anwendungsübergreifend zu unterstützen. Beispielsweise sollten sogenannte *Erweiterungsfelder* in bestehenden Protokollen oder Speichersystemen genutzt werden, um diese Sicherheitsklassifizierung mitzuteilen.

Das Modell der Sicherheitsklassifizierung korreliert eng mit den genutzten Sicherheitsmodellen (engl. Security Ontologies) der Infrastruktur für kontextsensitive Sicherheit (s. [FK11]). Das kann eine einfache hierarchische Einteilung beispielsweise mittels eines Schlüssel-Wert-Paars sein:

- $SecLevelClass = 0 \ldots 9$,

eine Verschlagwortung [EPS10] oder eine mehrdimensionale Informationssicherheitsziel-Angabe:

- $SecLevelClass =$ (Vertraulichkeit, Integrität, Verfügbarkeit)
- $SecLevelClass = (2, 16, 1)$

Die Assets als Objekte des Interesses sind vor allem über die Modellkomponente erreichbar und damit auch Ihre Sicherheitsklassifizierung.

```
1  <xs:complexType name="ValueType" mixed="true">
2   <xs:choice>
3    ...
4    <xs:element name="seclevelclass" type="ContextActivityAssetSecLevelClass" />
5    ...
6   </xs:choice>
7  </xs:complexType>
8
9  <xs:simpleType name="ContextActivityAssetSecLevelClass">
10  <xs:restriction base="xs:int">
11   <xs:pattern value="0|1|2|3|4|5|6|7|8|9" />
```

```
12   </xs:restriction>
13  </xs:simpleType>
```

Quelltext 7.6: Datentyp XML-*Schema* Definition für *SecLevelClass*

Aus Gründen der Kohärenz wird das jeweils gültige Modell der Sicherheitsklassifizierung auch als Bezugsmodell zur Bewertung des Sicherheitskontextes genutzt (s. Kap. 5.2.4) und somit zentral von der genutzten Infrastruktur für kontext-sensitive Sicherheit vorgegeben. Der Übersicht halber wird die exemplarisch vorgestellte hierarchische Einteilung als Vorgabe der Infrastruktur angenommen und modelliert. Für die Benutzeraktivitätsbenachrichtigung wird folgender Schlüsselname verwendet:

- ACTIVITY_ASSET

Wie im Quelltext 7.6 dargestellt wird zusätzlich das XML-*Schema* um einen weiteren Datentyp *SecLevelClass* erweitert. Wobei die Wertigkeit mit den Informationssicherheitsanforderungen korreliert (Security Policy).

#### Kommunikationskanäle

Für die Modellierung des Kontextattributes *Channel* soll eine Klassifizierung zur Verfügung gestellt werden, welche die relevanten Kommunikationskanäle gruppiert. Dies soll den Umstand des breiten Spektrums an Schnittstellen[9] adressieren. Trotz einer grossen Schnittmenge an angebotenen Kommunikationskanälen der unterschiedlichen Soft- und Hardware-Komponenten empfiehlt sich eine entsprechende Generalisierung.

Kommunikationskanäle, über die Informationen fliessen, haben eine direktionale Komponente. Moderne Schnittstellen nutzen *Protokolle* für die Kommunikation über spezifische Kanäle. Diese Protokolle sind zu einem grossen Teil bidirektional konzipiert. Die Kommunikation über solche Kanäle kann demgegenüber wiederum primär unidirektional sein. Letzteres spiegelt eine höhere Abstraktionsebene wieder und soll hier aufgegriffen und zusätzlich zu

[9] Kommunikationskanäle resultieren aus der Nutzung von Schnittstellen

der Klassifizierung durch eine Typisierung modelliert werden. Die Typisierung spiegelt die direktionale Ausprägung explizit wieder. Aus der Perspektive des mobilen Endgeräts wird dazu die folgende Kennzeichnug genutzt:

- *Input* - Kanalnutzung bei der die Kommunikation primär aus dem Dateneingang besteht
- *Output* - Kanalnutzung bei der die Kommunikation primär aus dem Datenausgang besteht
- *InputOutput* - Kanalnutzung bei der die Kommunikation primär bidirektional statt findet

Bezüglich der Klassifizierung der Kommunikationskanäle werden folgende Dimensionen vorgestellt, die auf einer höheren Abstraktionsebene relevante Kontextinformationen bereitstellen (zur Erläuterung werden Beispiele beigefügt):

- *Internal* - Kanäle die für den anwendungs-internen Datenaustausch genutzt werden, und indirekt beispielsweise aus *Local*-Kanäle resultieren (oder zu solchen führen)
- *Local* - Beispielsweise für die grafische Ausgabe (Display) oder einer anwendungs-übergreifenden *Cut/Paste*-Funktion[10]
- *ExternalLocal* - Kanäle die eine externe Verbindung repräsentieren; HID, USB, SD-Card-Bus, NFC
- *ExternalWide* - Kanäle die eine grössere externe Reichweite aufweisen; Audio-Output, HDMI, Bluetooth, WLAN, Audio-Input: VoiceInteraction (IPA)
- *Remote* - Kanäle die einen entfernten Datenaustausch ermöglichen; HTTP, TCP-Sockets, LTE/3G (VoiceCall), GPS

[10] Als GUI-Kanal für den Datenaustausch hat dieser Zwischenspeicher anwendungsübergreifende Auswirkungen, daher wird dieser *Kanal* anstatt in *Internal* in *Local* eingeordnet.

Die Beziehungen zwischen den Dimensionen können bereits implizit aus der hier getroffenen Wahl der Dimensionen abgeleitet werden. Dennoch soll die Modellierung dieser Beziehungen beispielsweise durch eine Entscheidung, ob diese Beziehungen über eine Inklusions- oder einer Komplementvorschrift besteht, explizit dem eingesetzten Sicherheitsmodell überlassen werden (Security Policy).

```
<xs:complexType name="ValueType" mixed="true">
 <xs:choice>
...<xs:element name="channel" type="ContextActivityChannel" />...
 </xs:choice>
</xs:complexType>

<xs:complexType name="ContextActivityChannel">
 <xs:sequence>
  <xs:element name="class" type="ContextActivityChannelClass" />
  <xs:element name="type" type="ContextActivityChannelType" />
 </xs:sequence>
</xs:complexType>

<xs:simpleType name="ContextActivityChannelType">
 <xs:restriction base="ASCIIString">
  <xs:pattern value="input|output|inputoutput" />
 </xs:restriction>
</xs:simpleType>

<xs:simpleType name="ContextActivityChannelClass">
 <xs:restriction base="ASCIIString">
  <xs:pattern value="internal|local|externallocal|externalwide|remote" />
 </xs:restriction>
</xs:simpleType>
```

Quelltext 7.7: Datentyp XML-*Schema* Definition für *Channel*

Mit der Klassifizierung und der Typisierung werden die Kontextinformationen des Kontextattributs *Channel* als Tupel modelliert. Wie im Quelltext 7.7 dargestellt, wird dazu ein weiterer zweiwertiger Datentyp *Channel* dem XML-*Schema* hinzugefügt. Für die Benutzeraktivitätsbenachrichtigung wird folgender Schlüsselname verwendet:

- ACTIVITY_CHANNEL

Hinsichtlich der Anwendungsarchitektur ist die Steuerungskomponente (MVC) primär für alle Kommunikationskanäle der Benutzerschnittstelle verantwortlich (GUI/HID). Während die Nutzung anderer Kommunikationskanäle vor allem in der Modellkomponente (MVC) identifiziert wird.

### Datenstruktur

Anhand der in den vorangegangenen Abschnitten detailiert erläuterten Kontextattribute lässt sich eine spezifische sicherheitsrelevante Benutzeraktivität beispielsweise wie folgt formal beschreiben (erweitertes XML-RPC Datenmodell):

```
1   ...
2     <value>
3       <struct>
4           <member>
5 <!-- Linear security classification 0-9 -->
6               <name>ACTIVITY_ASSET</name>
7                 <value>
8                     <seclevelclass>8</seclevelclass>
9                 </value>
10              </member>
11          <member>
12 <!-- Activity operation -->
13              <name>ACTIVITY_OPERATION</name>
14                  <value>
15                      <operation>read</operation>
16                      </value>
17              </member>
18          <member>
19 <!-- Activity channel -->
20              <name>ACTIVITY_CHANNEL</name>
21                  <value>
22                    <channel>
23                        <class>externalwide</class>
24                        <type>output</type>
25                    </channel>
26                  </value>
27              </member>
28          </struct>
29        </value>
30  ...
```

Quelltext 7.8: Sicherheitsrelevante Benutzeraktivität

Die einzelnen Kontextattribute konstituieren in Ihrer Summe die Benutzeraktivität (Quelltext 7.8), aber *nicht* vollständig den Anwendungskontext. Für die Kommunikation des Anwendungskontextes $AppContext$ wie es im Kapitel 5.2.4 vorgestellt wird, bedarf es allgemein noch des korrespondieren Anwendungszustands. Insbesondere die aktive Sicherheitsmassnahmenkonfiguration $Capabilities_A$, dass die entsprechende sicherheitsrelevante Benutzeraktivität kontextuiert.

In den vorangegangenen Abschnitten wird ein generisches Anwendungskontextmodell zur Modellierung sicherheitsrelevanter Benutzeraktivitäten und ihre anwendungs-architektonische Verortung vorgestellt. Die Modellierung der jeweiligen Kontextattribute als einzelne *Datentypen* hat den Vorteil, dass Ihre Werte bereits vor der Übermittlung des Anwendungskontextes bezüglich eines gültigen Wertebereiches verifiziert werden können. Somit werden fehlerhafte Kontextinformationen identifiziert. Für die Vervollständigung des Anwendungskontextes wird im nächsten Abschnitt das Anwendungsadaptionsmodell vorgestellt, dass unteranderem die Sicherheitsmassnahmenkonfiguration $Capabilities_A$ vorstellt.

### 7.3.2. Anwendungsadaptionsmodell

#### Homogenität

Die kontext-sensitive Sicherheitsadaption wird von der einzelnen Anwendung durchgeführt. Diese Durchführung von Sicherheitsmassnahmen impliziert, dass die Auswahl an Sicherheitsmassnahmen, die durchgeführt werden sollen und dementsprechend auch die Menge an zur Verfügung stehenden Sicherheitsmassnahmen, im Vorfeld dem Kontextserver und der Anwendungskomponente bekannt sind. Beiden Architekturkomponenten wird somit ein Konsens unterstellt — dies ist natürlich mitnichten der Fall. Um dieser Forderung nachzukommen, kann eine gegenseitige Abbildung der Mengen von Sicherheitsmassnahmen vorgenommen werden, die jeweils von der Anwendungskomponente und dem Kontextserver bereitgestellt werden [Kob12]. Solch eine Abbildung weist allerdings aufgrund der Heterogenität von Massnahmen Konsistenzprobleme auf. Wie im Kapitel 5.2.2 vorgestellt, wird der Konsens über

eine Menge an Sicherheitsmassnahmen hergestellt, indem die Anwendungskomponente diese Menge vorgibt (s. a. Kap. 5.2.3). Somit wird von vornherein sichergestellt, dass die Auswahl an Sicherheitsmassnahmen, die durchgeführt werden sollen, auch von der Anwendungskomponente *vollumfänglich* durchgesetzt werden können.

#### Angriffsflächenreduzierung

Die im Kapitel 5.2.2 vorgestellte Herangehensweise bezüglich der Sicherstellung der Informationssicherheit der Assets, sieht Sicherheitsmassnahmen in Form einer Angriffsflächenreduzierung vor. Eine Anwendungsangriffsfläche ($SecAppSurface$) ist demnach wie folgt definiert:

$$SecAppSurface \subsetneq Assets \cup Operations \cup Channels \tag{7.1}$$

Die Anwendungsangriffsfläche wird von drei Interaktionsmerkmalen beschrieben. Das Assets-Kontextattribut stellt Attributwerte zur Verfügung, die primär die möglichen Schutzziele vorgeben. Faktisch wird in dieser Arbeit diese Dimension ausschliesslich dafür eingesetzt[11]. Das Operations-Kontextattribut stellt eine Einzelmenge zur Verfügung, das alle asset-betreffenden Aktionen repräsentiert. Das Channels-Kontextattribut stellt abschliessend eine Tupelmenge zur Verfügung, die die klassifizierende Taxonomie der Kommunikationskanäle repräsentiert. Die Kontextattribute stellen jeweils als Tupelmenge oder Einzelmenge das Spektrum an Anwendungskontextinformationen zur Verfügung und bilden zusammen eine Attribut- respektive Wertemenge (s. Kap. 7.3.1). Die Angriffsflächenreduzierung nimmt somit eine Reduzierung dieser *SecAppSurface* Menge vor. Die Wertemenge reflektiert hierbei die Abstraktion und steht repräsentativ für die referenzierten Objekte der Interaktionen. Wird ein Wertebereich eingeschränkt so stehen die Objekte nicht zur Verfügung, die Attribute mit Werten aufführen, die ausserhalb des Wertebereiches liegen. Mit

$$\Delta SecAppSurface = SecAppSurface_{k_{t_2}} - SecAppSurface_{k_{t_1}} \tag{7.2}$$

[11] Eine Erweiterungsmöglichkeit beispielsweise für eine Typisierung [EPS10] bleibt bei dieser Modellierung intentional erhalten.

stellt $\Delta SecAppSurface$ als Teilmenge selbst, die de facto Sicherheitsmassnahme dar[12]. Da solch eine Sicherheitsmassnahme anhand einer konkreten Angriffsflächenreduzierung anwendungs-spezifische Sicherheitsoperationen $(SecOperations)$ benötigt, muss hier ebenso eine Abstraktion vorgenommen werden. Um die Ausführung respektive die Durchsetzung zu kapseln, wird wie in den vorangegangenen Abschnitten genutzt, die entsprechend in Kraft tretende oder kommunizierte Sicherheitsmassnahme als zur Verfügung stehende *Fähigkeiten* modelliert (engl. $Capabilities$, s. Kap. 5.2.3). In der Regel impliziert dies die vollständige Adaptionsfähigkeit der Anwendungsangriffsfläche. Zusätzlich zu dieser Abstraktion spiegelt diese Sichtweise folgendes Sicherheitsmodell wieder: in Anlehnung an ein *Whitelist*-Ansatz werden Sicherheitsmassnahmen stets als *erlaubte* Teilmenge der Fähigkeiten der Anwendung kommuniziert und *nicht* invertiert in Form einer *nicht-erlaubten* Teilmenge wie sie $\Delta SecAppSurface$ beschreibt. Ersteres spiegelt nicht nur die erlaubte Angriffsfläche als Sicherheitsmassnahme wieder, sondern ist auch Aufgrund der expliziten Entscheidung verlässlicher und transparenter bezüglich der Informationssicherheitsziele. Die Angriffsflächenadaption als Sicherheitsmassnahme wird somit in Form von erlaubten Teilmengen der Anwendungsangriffsfläche respektive der Anwendungsfähigkeiten $(Capabilities)$ formuliert.

#### Sicherheitsmassnahmenkonfiguration

Die während einer aktuellen Benutzeraktivität aktive $(Capabilities_A)$ oder als neue Sicherheitsmassnahme $(Capabilities_m)$ kommunizierte Teilmenge der Anwendungsfähigkeit (s. Kap. 5.2.4) stellt unterschiedliche Konfigurationen dar und soll allgemein als Sicherheitsmassnahmenkonfiguration verstanden werden $(Capabilities)$. Für die Modellierung wird das in den vorangegangenen Kapiteln vorgestellte und erweiterte XML-RPC Datenmodell eingesetzt (s. a. Anhang A.1.1). Aus den konzeptuellen Überlegungen (s. Kap. 5) lassen sich drei Sicherheitsmassnahmenkonfigurationen identifizieren. Diese werden mit folgenden Schlüsselnamen modelliert:

[12] Die Adaption stellt ebenso Objekte wieder zur Verfügung (s. Kap. 5.2.4)

- CAPABILITIES_APP
  Vollständige Anwendungsfähigkeiten (Anwendungsangriffsfläche)
- CAPABILITIES_ACTIV
  Aktive Teilmenge der Anwendungsfähigkeit (aktive Sicherheitsmassnahme)
- CAPABILITIES_MEASURES
  Neue Teilmenge der Anwendungsfähigkeit (neue Sicherheitsmassnahme)

Um einen Konsens über mögliche Sicherheitsmassnahmenkonfigurationen herstellen zu können, muss die gesamte Anwendungsfähigkeit ($Capabilities_{App}$) der Infrastruktur für kontext-sensitive Sicherheit zur Verfügung gestellt werden (s. Kap. 7.1.1 und Abb. 7.2). Anhand der im vorangegangenen Kapitel 7.3.1 detailiert erläuterten Interaktionsmerkmale lässt sich darauf aufbauend die sicherheitsrelevanten Anwendungsfähigkeiten (CAPABILITIES_APP) wie folgt formal beschreiben (XML-RPC Datenmodell):

```
1  ...
2  <!-- Application capabilities -->
3  <name>CAPABILITIES_APP</name>
4  <value>
5  <struct>
6  <member>
7  <!-- Linear class; maximum asset security class enough (scope 0-max) -->
8  <name>ATTRIBUTE_ASSET</name>
9  <value>
10   <array>
11     <data>
12       <value><seclevelclass>9</seclevelclass></value>
13      </data>
14    </array>
15 </value>
16 </member>
17 <member>
18 <!-- All operations -->
19 <name>ATTRIBUTE_OPERATION</name>
20 <value>
21   <array>
22     <data>
23       <value><operation>attribute</operation></value>
24       <value><operation>close</operation></value>
25       <value><operation>create</operation></value>
```

```
26          <value><operation>delete</operation></value>
27          <value><operation>move</operation></value>
28          <value><operation>modify</operation></value>
29          <value><operation>open</operation></value>
30          <value><operation>read</operation></value>
31          <value><operation>write</operation></value>
32        </data>
33      </array>
34  </value>
35  </member>
36  <member>
37  <!-- All channels -->
38  <name>ATTRIBUTE_CHANNEL</name>
39  <value>
40     <array>
41        <data>
42          <value>
43            <channel><class>internal</class><type>input</type></channel>
44            </value>
45          <value>
46            <channel><class>internal</class><type>output</type></channel>
47            </value>
48          <value>
49            <channel><class>internal</class><type>inputoutput</type></channel>
50            </value>
51          <value>
52            <channel><class>local</class><type>input</type></channel>
53            </value>
54          <value>
55            <channel><class>local</class><type>output</type></channel>
56            </value>
57          <value>
58            <channel><class>local</class><type>inputoutput</type></channel>
59            </value>
60          <value>
61            <channel><class>externallocal</class><type>input</type></channel>
62            </value>
63          <value>
64            <channel><class>externallocal</class><type>output</type></channel>
65            </value>
66          <value>
67            <channel><class>externallocal</class><type>inputoutput</type></channel>
68            </value>
69          <value>
70            <channel><class>externalwide</class><type>input</type></channel>
71            </value>
72          <value>
73            <channel><class>externalwide</class><type>output</type></channel>
74            </value>
75          <value>
76            <channel><class>externalwide</class><type>inputoutput</type></channel>
```

```
77            </value>
78          <value>
79            <channel><class>remote</class><type>input</type></channel>
80            </value>
81          <value>
82            <channel><class>remote</class><type>output</type></channel>
83            </value>
84          <value>
85            <channel><class>remote</class><type>inputoutput</type></channel>
86            </value>
87        </data>
88     </array>
89 </value>
90 </member>
91 </struct>
92 </value>
93 ...
```

Quelltext 7.9: Spektrum an Anwendungsfähigkeit ($Capabilities_{App}$)

Wie im Quelltext 7.9 einzusehen, wird das Spektrum an Anwendungsfähigkeiten als Verbund modelliert. Bezeichnet mit CAPABILITIES_APP beinhaltet es die einzelnen Kontextattributmengen als *Array* mit folgenden dedizierten Schlüsselnamen:

- ATTRIBUTE_ASSET
- ATTRIBUTE_OPERATION
- ATTRIBUTE_CHANNEL

Die im Quelltext 7.9 modellierten Anwendungsfähigkeiten können auch vom XML-*Schema* beziehungsweise vom XML-RPC Datenmodell abgeleitet werden, da der Quelltext 7.9 den vollen Umfang beinhaltet. Allerdings muss davon ausgegangen werden, dass nicht alle Anwendungen diese Umfang anbieten und somit eine explizite Kommunikation des Spektrums an Anwendungsfähigkeiten unumgänglich ist.

Auf Basis dieser sicherheitsrelevanten Anwendungsfähigkeiten kann eine Infrastruktur für kontext-sensitive Sicherheit eine Sicherheitsmassnahme in Form einer Sicherheitsmassnahmenkonfiguration kommunizieren. Beispielsweise

kann eine mögliche Sicherheitsmassnahmenkonfiguration (CAPABILITIES-_MEASURES), bei der aufgrund des bewerteten Sicherheitskontextes manipulative Operationen nicht erlaubt und zusätzlich die Nutzung externer Kommunikationskanäle verweigert werden sollen, wie folgt kommuniziert werden (XML-RPC Datenmodell):

```
...
<!-- Measures for Application (Capabilities_m) -->
<name>CAPABILITIES_MEASURES</name>
 <value>
  <struct>
   <member>
<!-- Security classification correspondes with activity class -->
    <name>ATTRIBUTE_ASSET</name>
        <value>
           <array>
              <data>
                <value><seclevelclass>5</seclevelclass></value>
               </data>
             </array>
        </value>
    </member>
   <member>
<!-- Write related operations are not listed-->
    <name>ATTRIBUTE_OPERATION</name>
        <value>
           <array>
              <data>
                <value><operation>close</operation></value>
                <value><operation>open</operation></value>
                <value><operation>read</operation></value>
               </data>
             </array>
        </value>
    </member>
   <member>
<!-- No remote, externallocal and externalwide channels allowed -->
    <name>ATTRIBUTE_CHANNEL</name>
       <value>
         <array>
           <data>
             <value>
              <channel><class>internal</class><type>input</type></channel>
              </value>
             <value>
              <channel><class>internal</class><type>output</type></channel>
              </value>
             <value>
              <channel><class>internal</class><type>inputoutput</type></channel>
```

```
44                </value>
45               <value>
46                <channel><class>local</class><type>input</type></channel>
47                </value>
48               <value>
49                <channel><class>local</class><type>output</type></channel>
50                </value>
51               <value>
52                <channel><class>local</class><type>inputoutput</type></channel>
53                </value>
54             </data>
55           </array>
56         </value>
57      </member>
58    </struct>
59  </value>
60 ...
```

Quelltext 7.10: Sicherheitsmassnahmenkonfiguration $(Capabilities_m)$

Wie im Quelltext 7.10 einzusehen, bleibt die Datenstruktur für Sicherheitsmassnahmenkonfigurationen im Vergleich zum Spektrum an Anwendungsfähigkeit erhalten. Bezeichnet mit CAPABILITIES_MEASURES beinhaltet es die einzelnen adaptierten Kontextattributmengen als *Array* mit den gleichen dedizierten Schlüsselnamen:

- ATTRIBUTE_ASSET
- ATTRIBUTE_OPERATION
- ATTRIBUTE_CHANNEL

Bezüglich der Datenstruktur gilt, dass alle Sicherheitsmassnahmenkonfigurationen die gleiche Strukturierung aufweisen. Lediglich deren Schlüsselnamen unterscheiden sich (CAPABILITIES...).

Für die Sicherheitsmassnahmenkonfiguration CAPABILITIES_ACTIV, die zusammen mit den Kontextattributen als Anwendungskontext (s. Kap. 5.2.4) von der Anwendung an das Kontextrahmenwerk kommuniziert wird und somit die sicherheitsrelevante Benutzeraktivität kontextuiert $(Capabilities_A)$, gilt, dass

diese mit einer als Sicherheitsmassnahme kommunizierten Sicherheitsmassnahmenkonfiguration CAPABILITIES_MEASURES identisch ist. Allerdings können unterschiedliche Sicherheitsmassnahmenkonfigurationen in unterschiedlichen *View*-Instanzen der Präsentationskomponente (MVC) aktiv sein und somit kann nicht davon ausgegangen werden, dass die *zuletzt* angewendete Sicherheitsmassnahmenkonfiguration CAPABILITIES_MEASURES der aktuell aktiven Sicherheitsmassnahmenkonfiguration CAPABILITIES_ACTIV entspricht.

#### Sicherheitsoperation

Das auf dem Kontextserver zum Einsatz kommende Sicherheitsmodell entscheidet über die Sicherheitsmassnahme respektive Sicherheitsmassnahmenkonfiguration, um den Informationssicherheitszielen zu genügen (s. [MB04]). Dementsprechend wird kontext-sensitiv über das *Was* zu tun ist entschieden. Die bisherige Modellierung der Sicherheitsmassnahme macht keine explizite Angabe über die Art und Weise[13] der Durchsetzung in der Anwendung $(SecOperations)$. Die implizite Annahme, dass die Sicherheitsmassnahme *Angriffsflächenreduzierung* vergleichbar mit einer klassischen Zugriffssteuerung ist (Autorisierungsüberprüfung), ist hinfällig. Insbesondere da es für eine kontextsensitive Adaption nicht flexibel genug ist [CCB08]. Zusätzlich sind die Sicherheitsmassnahmen in der Benutzeranwendung verortet, was wiederum weitere Anforderungen für die Interaktionsgestaltung stellt [HM06, DIN06]. Daher ist es essentiell, dass neben einer Autorisierungsüberprüfung weitere Mechanismen angeboten werden, um somit die Sicherheitsmassnahme gestalten zu können. Folgende Mechanismen sollen dazu vorgestellt werden:

- *Authorization* (obligatorisch) - verweigert denn Zugriff auf $\Delta SecAppSurface$[14] (Autorisierungsüberprüfung)
- *Information* - verweigert denn Zugriff auf $\Delta SecAppSurface$ und gibt UI Feedback

[13] nicht zu verwechseln mit; *Wie* die Ausführung genau gestaltet ist.
[14] siehe (7.2) auf Seite 55.

- *Alternative* - verweigert denn Zugriff auf $\Delta SecAppSurface$ und stellt Alternativen zur Verfügung

Ebenso kann eine Aufhebung der Sicherheitsmassnahme erlaubt werden, wenn dies im Rahmen der Interaktionsgestaltung als sinnvoll erachtet wird (Steuerbarkeit [DIN06]):

- *Confirmation* - erlaubt denn Zugriff auf $\Delta SecAppSurface$ durch Einholung eines Benutzereinverständnisses
- *Authentication* - erlaubt denn Zugriff auf $\Delta SecAppSurface$ durch Authentifizierung

Somit können neben der Sicherheitsmassnahmenkonfiguration Informationen zusätzlich vom Kontextserver kommuniziert werden, die es der Sicherheitsoperation in der Anwendung erlaubt die Sicherheitsmassnahme flexibler auszuführen. Die Flexibilität wird dadurch erreicht, dass mehrere erlaubte Mechanismen kommuniziert werden können, um der Sicherheitsoperation in der Anwendung mehrere Möglichkeiten zu erlauben. Die Anwendungskomponente (Security Component) muss für jede zur Verfügung stehende Sicherheitsoperation mindestens den Mechanismus der Autorisierungsüberprüfung implementieren. Hat eine Sicherheitsoperation einen kommunizierten *optionalen* Mechanismus nicht implementiert, so gilt stets die Autorisierungsüberprüfung als Standard.

```
1  <xs:complexType name="ValueType" mixed="true">
2   <xs:choice>
3    ...
4    <xs:element name="mechanism" type="MeasuresMechanism" />
5   </xs:choice>
6  </xs:complexType>
7
8  <xs:simpleType name="MeasuresMechanism">
9   <xs:restriction base="ASCIIString">
10   <xs:pattern
11    value="authorization|information|alternative|confirmation|authentication" />
12  </xs:restriction>
13 </xs:simpleType>
```

Quelltext 7.11: Datentyp XML-*Schema* Definition für *Mechanism*

Für die Modellierung wird wie im Quelltext 7.11 dargestellt ein neuer Datentyp definiert, der das Datenmodell aus Kapitel 7.1.1 erweitert. Für die Sicherheitsmassnahmenbenachrichtigung wird folgender Schlüsselname verwendet:

- MEASURES_MECHANISM

Für die Sicherheitsmassnahmenbenachrichtigung bedeutet dies, dass zusätzlich zu der Sicherheitsmassnahmenkonfiguration (Quelltext 7.10) die erlaubten Mechanismen beispielsweise wie folgt inkludiert werden:

```
1  ...
2  <!-- Example of allowed measures mechanism -->
3  <name>MEASURES_MECHANISM</name>
4      <value>
5          <array>
6              <data>
7                  <value><mechanism>alternative</mechanism></value>
8                  <value><mechanism>authorization</mechanism></value>
9              </data>
10         </array>
11     </value>
12 ...
```

Quelltext 7.12: Benachrichtigungsbestandteil für die Mechanismen

Die Anwendungskomponente (Security Component) wird anhand der Sicherheitsmassnahmenbenachrichtigung (partiell im Quelltext 7.12) die Variante der Sicherheitsoperation ausführen, die alternative Vorschläge zu dem Objekt macht, dass durch die Sicherheitsmassnahmen nicht erlaubt wird. Dies wird dem Benutzer dann priorisierte Alternativen zur Verfügung stellen.

Hinsichtlich der Anwendungsarchitektur (MVC) muss die adaptive Sicherheitsfunktionalität vollständigen Zugriff auf die Anwendungslogik haben, um zielgerichtet die Sicherheitsoperationen ausführen zu können. In Anbetracht der vorgestellten Mechanismen besteht eine starke Wechselwirkung zwischen der Steuerungskomponente und der Modellkomponente. Die entsprechende Interaktion muss daher von der Anwendungskomponente (Security Component) in beiden MVC-Komponenten implementiert werden.

#### Konfliktbehandlung

In der Regel werden unterschiedliche Datenobjekte entsprechend durch unterschiedliche *View*-Instanzen der Präsentationskomponente (MVC) repräsentiert. Diese können unabhängig von einander unterschiedlichen Sicherheitsmassnahmen unterliegen. Bei konkurierenden Adaptionsaktivitäten respektive der Durchsetzung von Sicherheitsmassnahmen gilt allerdings, dass die Assets mit hohen Sicherheitsinformationszielen und deren Sicherheitsmassnahmen[15] nicht von Sicherheitsmassnahmen überschrieben werden können, die für Benutzeraktivitäten gelten, die auf Assets mit geringerem Schutzbedarf angewendet werden. Wird der Zugriff auf Assets mit hohem Schutzbedarf beendet (Close-Operation), so wird dies protokolliert und nicht weiter als massgebend bewertet.

Um dies sicherzustellen gilt, dass Sicherheitsmassnahmen einer Priorisierung unterliegen die mit der entsprechenden Benutzeraktivität respektive der zugehörigen Sicherheitsklassifizierung korrespondiert:

$$SecMeasuresPriority = |SecLevelClass| \tag{7.3}$$

Für die Sicherheitsmassnahmenbenachrichtigung wird folgender Schlüsselname verwendet:

- MEASURES_PRIORITY

und den Metadaten der Sicherheitsmassnahmen beigefügt (s. Kap. 7.1.2). Für die Wertzuweisung erübrigt sich eine Erweiterung des XML-*Schema*, da der Datentyp *SecLevelClass* der Asset-Klassifizierung hierfür genutzt wird und bereits zur Verfügung steht (Quelltext 7.6).

In den vorangegangenen Abschnitten wird ein generisches Anwendungsadaptionsmodell für die Modellierung der adaptiven Sicherheitsfunktionalität und ihre anwendungs-architektonische Verortung vorgestellt. Diese Verortung soll im nächsten Abschnitt der Übersicht halber nochmals aufgegriffen werden.

[15] hohe Sicherheitsinformationsziele impliziert restriktivere Sicherheitsmassnahmen

### 7.3.3. Integration der Anwendungskomponente

Die Anwendungskomponente (Security Component) der einzelnen Anwendungen muss neben der Integration adaptiver Sicherheitsfunktionalitäten ebenso die Ermittlung von sicherheitsrelevanten Benutzeraktivitäten in der Anwendung durchführen. Die Benutzeraktivität steht im direkten Dialog mit der Steuerungskomponente (MVC), die gleichzeitig die Anwendungslogik maskiert. Dem gegenüber steht das Interesse an den Assets mit ihren Sicherheitsklassifizierungen, die vor allem über die Modellkomponente (MVC) erreichbar sind. Ebenso finden sicherheitsrelevante Operationen entkoppelt von der Benutzerschnittstelle in der Modellkomponente statt. Das Interaktionsmerkmal *Kommunikationskanäle* zeigt insbesondere den kausalen Zusammenhang auf, dass sensitive Informationen sowohl die Steuerungskomponente als auch die Modellkomponente durchlaufen. Allein aus dieser Betrachtung der Identifikation sicherheitsrelevanter Benutzeraktivität empfiehlt sich, die Anwendungskomponente als eine Querschnittskomponente einer Anwendungsarchitektur zu betrachten. Mit Blick auf die adaptive Sicherheitsfunktionalität wird diese Quer-

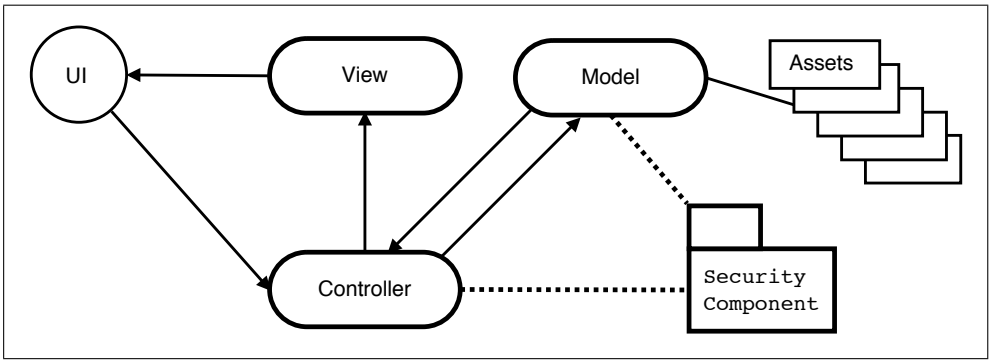


Abb. 7.4.: Schematische Darstellung der Integration der Anwendungskomponente (Security Component)

schnittsrole weiter untermauert. Für die zielgerichtete Ausführung der Sicherheitsoperationen ist die Integration in die Modellkomponente notwendig. Beispielsweise muss auch eine zu kommunizierende Benutzeraktivität stets in der aktiven Sicherheitsmassnahmenkonfiguration identifiziert werden. Des Weiteren geht die Sicherheitsfunktionalität ebenso mit einer Interaktionsgestaltung

einher, die zusätzlich die Steuerungskomponente einbindet.

Hinsichtlich der Verortung der Anwendungskomponente in die im Kapitel 7.2 vorgestellte Anwendungsarchitektur, zeigt sich, dass die entsprechende Funktionalität wie die kontext-sentive Adaption und die Kontextquelle Benutzeraktivität, nur durch eine umfassende Unterstützung der Anwendungsarchitektur ermöglicht wird. Als Querschnittskomponente ist die Anwendungskomponente (Security Component) insbesondere in der Modell- und Steuerungskomponente zu integrieren (s. Abb. 7.4).

## 7.4. Zusammenfassung

In den vorangegangenen Abschnitten werden verschiedene signifikante Modelle vorgestellt, die zusammen die Integration des kontext-adaptiven Verhaltens inklusive der Kommunikation von sicherheitsrelevanter Benutzeraktivität in Benutzeranwendungen ermöglichen (Anwendungskomponente). Ebenso wird die Anbindung von Benutzeranwendungen an die vorgestellte Infrastruktur für kontext-sensitive Sicherheit erläutert (Interaktionsmodell). Im ersten Abschnitt wird insbesondere mit dem Kommunikationsmodell die Struktur vorgestellt, die den Rahmen für die Kommunikation der Benutzeraktivität und für die Kommunikation der Sicherheitsmassnahmen bildet. Im letzten Abschnitt werden die Kontext- und Adaptionsmodelle vorgestellt, welche die eigentliche Benutzeraktivität in Form von Anwendungskontextinformationen und die Sicherheitsmassnahmen in Form einer Sicherheitsmassnahmenkonfiguration formal beschreiben. Ebenso wird der *Whitelist*-Ansatz bezüglich der Sicherheitsmassnahmen explizit dargelegt. Diese werden in Form von erlaubten Teilmengen der Anwendungsangriffsfläche respektive der Anwendungsfähigkeiten formuliert. Für die Integration der Anwendungskomponente (Security Component) wird für die Anwendungsentwicklung der Bezugsrahmen zum gängigen MVC Entwurfsmuster vorgestellt und die Anwendungskomponente als Querschnittskomponente darin verortet.

Die formale Modellierung der Kontextinformationen, Sicherheitsmassnahmen und der Kommunikation wird entlang des XML-RPC [Win03] Protokolls

[MSX16] in Form eines W3C XML-Schema [W3C12] vorgenommen. Zusätzlich werden anwendungskontext-spezifische Erweiterungen entwickelt und spezifiziert, und integriert im XML-RPC Datenmodell vorgestellt (s. Anhang A.1.1). Die Aufbereitung der Kontextinformationen unterstützt die Repräsentation verschiedener Datentypen[16]. Das vorgestellte Kommunikationsmodell sieht für die Kommunikation der sicherheitsrelevanten Benutzeraktivität (Anwendungskontextmodell) via $ActivityNotifyCb()$-Aufruf folgende Informationen konsolidiert vor:

- Sicherheitsrelevante Benutzeraktivität
  1. Benachrichtigungsmetadaten

     ACTIVITY_APPLICATION_ID

     ACTIVITY_UUID

     ACTIVITY_APP_TIMESTAMP
  2. Anwendungskontextinformation

     ACTIVITY_OPERATION

     ACTIVITY_ASSET

     ACTIVITY_CHANNEL

     CAPABILITIES_ACTIV

Für die anwendungs-spezifischen Sicherheitsmassnahmen, die korrespondierend zu den kommunizierten Benutzeraktivitäten von der Infrastruktur für kontext-sensitive Anwendungen bereit gestellt werden, sieht das vorgestellte Kommunikationsmodell und das Anwendungsadaptionsmodell für die Kommunikation der Sicherheitsmassnahmen via $SecResultNotifyCb()$-Aufruf folgende Informationen konsolidiert vor:

- Anwendungs-spezifische Sicherheitsmassnahmen
  1. Benachrichtigungsmetadaten

[16] siehe Anhang A.1.1 auf Seite 83

MEASURES_REF_ACTIVITY_UUID

MEASURES_PRIORITY

MEASURES_SECGOAL

2. Sicherheitsmassnahmenkonfiguration

MEASURES_MECHANISM

CAPABILITIES_MEASURES

Um die Infrastruktur für kontext-sensitive Anwendungen in die Lage zu versetzen anwendungs-spezifische Sicherheitsmassnahmen zu kommunizieren, sieht das vorgestellte Kommunikationsmodell und das Anwendungsadaptionsmodell einmalig während des Registrierungsvorgangs die Kommunikation der gesamten anwendungs-spezifischen Fähigkeiten via *PublicServiceChannelforMeasures*()-Aufruf wie folgt vor:

- Spektrum an Anwendungsfähigkeiten
  1. Benachrichtigungsmetadaten

     MEASURES_APPLICATION_ID

     MEASURES_CALLBACK
  2. Spezifische Anwendungsfähigkeiten

     CAPABILITIES_APP

Die Kommunikation der strukturierten Anwendungskontextinformationen stellt Anforderungen an die weiterverarbeitenden Infrastrukturkomponenten. Die Kontextmodelle und Sicherheitsmodelle, die von der Infrastruktur für kontext-sensitive Sicherheit angewendet werden, müssen das vorgestellte Anwendungskontextmodell und das Anwendungsadaptionsmodell vollumfänglich integrieren.

Vollständig instanziierte XML-RPC-Aufrufe für die Benachrichtigung bezüglich einer sicherheitsrelevanten Benutzeraktivität, einer spezifischen Sicherheitsmassnahme und der Anwendungsfähigkeiten stehen des Umfangs wegen im

Anhang A.1.2, A.1.3 und A.1.4 zur Verfügung. Die Validierung dieser Beispiele wurde mit Werkzeugen aus GNOME's *libxml2*-Paket[17] durchgeführt.

Um den vorgestellte Ansatz zur Einbeziehung sicherheitsrelevanter Benutzeraktivitäten und der Umsetzung von Sicherheitsmassnahmen etwas greifbarer zu erläutern, wird er im nächsten Abschnitt diskutiert und beispielhaft angewendet.

[17] http://xmlsoft.org/

# 8. Diskussion

In den vorangegangenen Kapiteln wird ein Ansatz zur Einbeziehung sicherheitsrelevanter Benutzeraktivitäten vorgestellt. Darüber hinaus wird die Anwendbarkeit des deskriptiven Modells durch die konzeptuelle Integration in eine spezifische Infrastruktur für kontext-sensitive Sicherheit [FK11] dargelegt. Um den prinzipiellen Einsatz weiter zu erläutern, werden verschiedene Aspekten bezüglich des vorgestellten Modells im Folgenden diskutiert. Dazu werden unter anderem Szenarien vorgestellt, in denen der Einsatz einer Infrastruktur für kontext-sensitive Sicherheit das entwickelte Modell zur Anwendung bringt. Die Szenarien beschreiben *Alice* in einem beruflichen Umfeld. Alice ist aktuell in verschiedene Akquiseaktivitäten involviert und steht somit im engen Kontakt mit bestehenden und neuen Kunden.

## 8.1. Anwendbarkeit

### 8.1.1. Sicherheitsmassnahmen

Die Herausforderung bei der Einbeziehung der heterogenen Anwendungsebene für einen kontext-sensitiven Sicherheitsdienst besteht darin, eine Massnahmenstruktur anzubieten, die mögliche Sicherheitsmassnahmen und ermittelte Massnahmen zusammenhängend beschreibt. Der vorgestellte Ansatz adressiert diese Forderung, indem prinzipiell davon ausgegangen wird, dass eine einsetzbare Menge an Sicherheitsmassnahmen jeweils anwendungs-spezifisch ist. Indem die einzelnen Anwendungen ihren Umfang der implementierten Funktionalität kommunizieren, wird die Infrastruktur für kontext-sensitive Sicherheit in die Lage versetzt, konsistente Sicherheitsmassnahmen zu ermitteln.

Diese Flexibilität erlaubt die effektive Sicherheitsmassnahmenausführung und die Bereitstellung eines zuverlässigen Sicherheitsdienstes.

Bezüglich der Bewertung einer Sicherheitszielerreichung zeigen bestehende Herangehensweisen [ABKS09, Kob12], eine technologie-zentrierte Beurteilung von Bedrohungslagen. Sicherheitseigenschaften werden dabei ausschliesslich auf Basis von charakteristischen Merkmalen dieser Eigenschaften beurteilt [JGK14]. Für Bedrohungslagen ist eine technische Schwachstelle hinreichend aber keine notwendige Voraussetzung. Eine strukturelle oder organisatorische Schwachstelle wird ebenso zu einer Bedrohungslage führen (s. Kap. 2.1.2). Dazu folgendes Szenario, dass sich Merkmale der vorgestellten Infrastruktur für kontext-sensitive Sicherheit bedient:

Szenario 1: „*Während Alice entspannt im Aussenbereich der Unternehmenszentrale mit ihrem mobilen Endgerät die letzten Ergebnisse eines Gespräches am Vormittag einarbeitet, kommt ihr die Idee einige Umstrukturierungen an den Projektdaten vorzunehmen. Diese Benutzeraktivität wird fortlaufend von der Datenanwendung an die Infrastruktur für kontext-sensitive Sicherheit ihres Arbeitgebers kommuniziert. Da sich Alice in einem vertrauenswürdigen Bereich aufhält, werden die Aktionen nicht weiter durch Massnahmen eingeschränkt. Allerdings werden dabei unteranderem auch Löschaktionen rekursiv vorgenommen. Aufgrund der Speicherung vorangegangener Aktionen (Context History) und der Sicherheitsklassifizierung der Daten werden vom zentralen Kontextdienst weitere Löschaktionen als eventuelle Unachtsamkeit bewertet. Nach der zehnten Löschaktion in Folge werden Sicherheitsmassnahmen an das mobile Endgerät kommuniziert. Zusätzlich wird der Sicherheitsoperation in der Anwendung die Authentizierung als Sicherheitsmechanismus erlaubt. Alice erhält daraufhin den Hinweis zusammen mit der Ortsangabe der Datenlöschung, dass es sich eventuell um ungewollte Löschaktionen handeln kann. Um fortzufahren wird Alice um eine Authentizierung gebeten. Aufgrund der Ortsangabe der Datenlöschung bricht Alice diesen Vorgang ab und stellt fest, dass sie sich in einer falschen Ordnerebene befand und somit im Begriff war einen Datensatz ungewollt vollständig zu löschen.*“

Die Sicherstellung der Informationsverfügbarkeit ist neben der Integrität und der Vertraulichkeit von Informationen ebenso von sicherheitsrelevanter Bedeutung. Das triviale Beispiel im 1. Szenario zeigt, dass ein kontext-sensitiver Sicherheitsdienst weit mehr leisten muss, als nur Sicherheitseingenschaften tech-

nischer Funktionen zu beachten. Die Einbeziehung sicherheitsrelevanter Benutzeraktivitäten trägt einen signifikanten Beitrag dazu bei, indem sie die Ermittlung eines ganzheitlicheren Sicherheitskontextes unterstützen.

### 8.1.2. Anwendungskontext vs. Benutzeraktivität

Die Zuverlässigkeit der Aussagekraft von Kontextinformationen ist der Schlüssel jedes kontext-sensitive Sicherheitsdienstes. Bezüglich der Benutzeraktivität stellen deren Kontextinformationen bereits akkurate, die tatsächlich die Aktivität repräsentierende Informationen dar. Messfehler, die beispielsweise bei Kontextinformationen auftreten die von Hardwaresensoren herrühren, müssen nicht zusätzlich adressiert werden.

Die vorgestellte Modellierung der Benutzeraktivität geht weiter als nur die Extraktion von Kontextinformationen vorzunehmen. Die Anwendungskomponente (Security Componente) kommuniziert das Verhalten des Benutzers in Form eines Anwendungskontextes. Neben den Interaktionsmerkmalen werden die aktuell aktiven Sicherheitsmassnahmen kommuniziert (s. Kap. 5). Die Bewertung der Sicherheitszielerreichung nutzt diese Information um die Adaption zu ermitteln.

### 8.1.3. Sicherheitsadaption

Szenario 2: *„Während einer Veranstaltung eines Berufsverbandes trifft Alice auf potentiell neue Partner. Um Bob einem interessierten Gesprächspartner ihre persönlichen Kontaktdaten zur Verfügung zu stellen, nutzt sie ihr geschäftliches mobiles Endgerät. Alice's Intention ist, Bob ihre sensiblen Kontaktdaten in einem vCard-Format*[1] *mittels einer drahtlosen Verbindung (Bluetooth) zur Verfügung zu stellen. Alice öffnet dazu ihre Adressbuchanwendung. Die Anwendung stellt unmittelbar eine Verbindung zur Infrastruktur für kontext-sensitive Sicherheit ihres Arbeitgebers her. Nach der Auswahl ihrer Kontaktdaten in der Anwendung führt Alice die Export-Funktion für diesen Eintrag aus. Die Anwendung kommuniziert diese Benutzeraktivität an die Infrastruktur für kontext-sensitive Sicherheit. Aufgrund des unbekannten Standorts*

[1] http://www.ietf.org/rfc/rfc2426.txt

*und der Sicherheitsklassifizierung der Kontaktdaten erhält die Anwendung als Sicherheitsmassnahme keinen Zugriff auf ExternalWide-Kommunikationskanäle*[2]*. Zusätzlich wird der Sicherheitsoperation in der Anwendung die Bereitstellung von Alternativen erlaubt. Alice wird an der grafischen Benutzerschnittstelle ein Auswahlelement angezeigt (Widget), dass die Bluetooth-Funktion ausgegraut hat und an erster Stelle eine optische Exportfunktion anzeigt. Alice wählt die Variante für einen optischen Export aus. Daraufhin wird ein zweidimensionaler Matrix-Barcode am Display*[3] *angezeigt. Bob hat in der Zeit sein mobiles Endgerät in die Hand genommen und scannt für die Kontaktdatenübertragung mit seiner Adressbuchanwendung den von Alice am Display zur Verfügung gestellten Barcode.*"

In Anbetracht der dynamischen Sicherheitsadaption sowie der damit verbundenen nicht-deterministischen (adaptiven) Reaktion des mobilen Endgeräts, ist es für die Benutzerakzeptanz von kontext-sensitivem Sicherheitsverhalten von besonderem Interesse, dass für die Interaktion mit dem Benutzer die Grundsätze der Dialoggestaltung [DIN06] ausdrücklich beachtet werden (wie im Szenario 2 dargelegt). Gerade die Erwartungskonformität einer Anwendung wird durch das kontext-adaptive Sicherheitsverhalten beinträchtigt. Eine Steuerbarkeit auf Seiten des Benutzers kann den nötigen Freiraum aufrechterhalten. Die Grundsätze unterstützen nicht nur den Benutzer im Sinne des mentalen Modells des Benutzers bezüglich der Sicherheitsadaption, sondern sie unterstützen zusätzlich ein flexibleres Sicherheitsverhalten der Anwendungen. Wie im 2. Szenario illustriert findet eine kontrollierte sicherheitsbezogene Steuerung des *Workflows* statt. Dies entspricht einer angemesseneren Bereitstellung eines Sicherheitsdienstes, der auch von den Sicherheitsmodellen Unterstützung finden sollte (durch geeignete Auswahl der Sicherheitsmechanismen).

Um einen unterstützenden Sicherheitsdienst zur Verfügung zu stellen, sollten von der Anwendungsentwicklung die Ausführungen im Abschnitt *Sicherheitsoperation* auf Seite 62 eine besondere Aufmerksamkeit erhalten und *Exceptions* aufgrund einer Sicherheitsadaption explizit durch alternative Programmabläufe behandelt werden. Weiterführende Überlegungen hierzu sind eine prin-

[2] siehe Kapitel 7.3.1 auf Seite 50
[3] *Local*-Kommunikationskanal

zipielle Unterstützung von Bibliotheken für grafische Benutzerschnittstellen (GUI), für Sicherheitsinteraktionsmodelle mit alternativen Benutzerführungspfaden.

## 8.2. Einschränkungen

### 8.2.1. Sicherheitsklassifizierung

Szenario 3: *„Alice trifft sich mit Bob, dem Teamleiter einer Unterabteilung, um sich über den Stand eines Projektes zu informieren. Während des Gespräches werden Details mittels Alice mobilen Endgeräts zu spezifischen Projektdokumenten diskutiert. Diese Dokumente befinden sich im Intranet, wofür Alice von der Infrastruktur für kontext-sensitive Sicherheit ihres Arbeitgebers einen Lesezugriff erlaubt bekommt. Nach dem aufschlussreichen Gespräch verlässt Alice die Unterabteilung in Richtung ihres Büros. Während Alice im Aufzug steht, wirft sie noch einmal einen Blick in die Projektdokumente. Während dessen wird vom kontext-sensitiven Sicherheitsdienst an die Anwendung eine Notifizierung mit neuen Sicherheitsmassnahmen kommuniziert. Die Sicherheitsoperation wird dazu angewiesen einen Informationsmechanismus auszuführen. Alice erhält daraufhin in ihrer Datenanwendung die Information, dass der Lesezugriff im aktuellen Kontext nicht weiter gestattet ist und die Anwendung schliesst das Dokument.“*

Das Beispiel zeigt die Triggerung aufgrund einer Lokationsänderung. Allerdings liegt der Fokus auf der konkreten Sicherheitsklassifizierung der Projektdokumente. In dieser Arbeit ist die Sicherheitsklassifizierung der Assets nicht ausführlich behandelt und als gegeben betrachtet worden. Wie im Kapitel 7.3.1 ab Seite 48 angeführt sollte das genutzte Klassifizierungsmodell von der genutzten Infrastruktur für kontext-sensitive Sicherheit vorgegeben werden. Der vorgestellte Ansatz stellt die strukturellen Rahmenbedingungen bereit, um im Zusammenhang mit der Kommunikation der Benutzeraktivität den Anwendungskontext bereit stellen zu können. Die Behandlung der Anreicherung der zu schützenden Assets mit entsprechenden Metadaten (Sicherheitseinstufung)

und die anwendungs-übergreifende Kommunikation dieser Sicherheitsklassifizierung (3. Szenario, Intranetdokument) ist nicht ausreichend modelliert worden.

### 8.2.2. Identifikation

Die vorgestellte Modellierung von Benutzeraktiväten setzt in Anlehnung an [MW11] die Kommunikation der Interaktionsmerkmale rudimentär um. Eine Verfeinerung der Modellierung wird im wesentlichen zu einer genaueren Abbildung der Benutzeraktivät beitragen. Insbesondere ist die Asset- und Channels-Identifikation nicht eingeflossen. Eine Identifikation wird in ersten Ansätzen dazu beitragen, dass anwendungsübergreifende Konflikte identifiziert werden können. Beispielsweise stellt die zeitgleiche Nutzung des Mikrofons durch Anwendungen ein potentielles Vertraulichkeitsproblem dar. Die Entscheidung ob bestimmte Kommunikationskanäle oder Assets zeitgleich von mehreren Anwendungen genutzt werden können, obliegt dem Kontextserver respektive dem eingesetzten Sicherheitsmodell (Security Policy). In diesem Zusammenhang unterstützt eine Identifikation von Kommunikationskanälen oder Assets die Sicherheitsadaptionentscheidung.

## 8.3. Weiterführende Aspekte

### 8.3.1. Architekturkomponente

Die Sicherstellung der Informationssicherheit ist effektiv, wenn diese auf der Ebene stattfindet, wo die Daten als Informationen interpretiert werden können (s. [SRC84]). Dies ist aus Benutzersicht in der Regel auf der Anwendungsebene der Fall. Aufgrund der heterogenen Anwendungsebene ist eine Abstraktionsebene, welche die Interpretationslogik anwendungübergreifend bereitstellt, nicht zielführend. Die Anwendungsanbindung muss analog zu anderen Kontextquellen (s. [Kob12, Mül10]) generalisiert werden und jede Anwendung muss

Ihre Funktionalität spezifisch darauf abbildend kapseln. Solch eine Anwendungskomponente (Security Component) (s. Kap. 6.1.1) bringt zusätzlich Flexibilität mit ein. Die Umsetzung auf Anwendungsebene erlaubt unterschiedlichen Informationsicherheitszielen gleichzeitig gerecht zu werden, da die Sicherheitsmassnahmen granular umgesetzt werden können: Während eine *View*-Instanz (MVC) lesend ein Dokument zur Verfügung stellt, kann eine andere *View*-Instanz dies lesend und schreibend erlauben.

### 8.3.2. Laufzeit

Die Sicherheitsdimension strukturiert das Vorhaben einer kontext-sensitiven Adaption signifikant um, sodass allgemeine Empfehlungen für kontext-sensitive Applikationen nicht direkt nutzbar sind. Beispielsweise können ortsabhängige Anwendungen Kontextinformationen *direkt* verwenden. Hingegen muss die sicherheitsrelevante Benutzeraktivität vorweg eine Beurteilung vornehmen lassen, da sie selbst mit den Informationssicherheitszielen der Assets eine Referenz im Sicherheitskontext darstellt. Im Hinblick auf die vorgestellte Infrastruktur für kontext-sensitive Anwendungen (s. Kap. 6) ergibt sich aus der softwaretechnischen Betrachtung dazu eine wichtige Forderung: da die entscheidenden Daten im *Softwarestack* auf einer hohen Abstraktionsebene angesiedelt sind (Anwendungsebene), müssen diese Daten mehrere Softwareschichten durchlaufen. Die Relevanz einer schlanken und performativen Kommunikation ist somit evident. Das vorgestellte Kommunikationsmodell (s. Kap. 7.1.2) unterstützt auf dem betreffenden Abschied (Anwendungskomponente $\leftrightarrow$ Kontextrahmenwerk) hierzu persistente Verbindungen für den Transport [FGM+99] und XML-RPC [Win03] für eine schlanke und performative Kommunikation. Weiterführende Performance-Bewertungen können nur an konkreten Implementierungen vorgenommen werden (s. [BGF+10]). Bezüglich eines zentralen ontologie-basierten Kontextservices [FK11] zeigen sich bei hohen Datenaufkommen bereits hohe Rechenkosten auf [Ay07].

## 8.4. Ausblick

Kontext-sensitive Dienste weisen ein breites Spektrum auf. Hinzu kommt die fortschreitenden Entwicklung bei mobilen Endgeräten/Systemen. Mit Blick auf die Konvergenz von Mobil- und Desktop-Systemen werden auch Konzepte von Desktop-Systemen weiter Einzug auf Mobilsysteme halten. Neben der heute als Standard geltenden *Multitasking*-Ausführungsumgebung werden beispielsweise *Multiuser*-Funktionen auf mobilen Endgeräten in Zukunft keine Besonderheit mehr darstellen. Das vorgestellte Anwendungskontextmodell geht von einer *Singleuser*-Nutzung des Endgeräts aus. Weiterführende Auseinandersetzungen bezüglich einer *Multiuser*-Nutzung stehen somit noch aus.

Im Rahmen der Modellierung des Anwendungskontextes ist stets von einer Benutzer-*Aktivität* ausgegangen worden. Die Berücksichtigung einer *Inaktivität* ist nicht in die Modellierung eingeflossen. Mit Blick auf die Entwicklungen bei Wearable-Computing-Lösungen zeigen diese ein Potenzial[4] auf, um die Benutzer-*Inaktivität* bezüglich des Anwendungskontextes zu modellieren.

Wie die Szenarien aufzeigen ist ein Sicherheitsdienst nur durch eine vollständige Interaktion aller Architekturkomponenten umsetzbar. Die Modellierung sicherheitsrelevanter Benutzeraktivitäten als Kontextquelle stellt einen vertikalen Ausschnitt einer Infrastruktur für kontext-sensitive Sicherheit dar. Die Einbeziehung der vorgestellten Modelle in die übergreifenden Sicherheits- und Kontextmodelle wird wiederum zu weiteren Anforderungen führen, die in die vorgestellten Anwendungskontext- und Adaptionsmodelle inkludieren werden müssen. Beispielsweise werden Sicherheitsontologien zusätzlich benötigte Kontextattribute kommunizieren.

Das mobile Endgerät wie es aktuell verstanden wird (Smartphone, Tablet), wird in Zukunft einer weiteren Fragmentierung unterliegen. In diesem Zusammenhang wird eine Infrastruktur für kontext-sensitive Sicherheit ebenso eine verstärkt verteilte Ausprägung bekommen. Für die Komposition einer Benutzeraktivität respektive eines Anwendungskontextes müssen die Architektur-

[4] Szenario: *„Während Alice ihr Büro verlässt und ihr mobiles Endgerät auf dem Schreibtisch liegen lässt, schliesst die Anwendung die sensitiven Dokumente (…)“*

komponenten Mechanismen wie beispielsweise eine Autorisierung und Authentisierung inkludieren. Nur so kann sichergestellt werden, dass ein Anwendungskontext zu einem integeren Sicherheitskontext beiträgt.

# 9. Zusammenfassung

In dieser Arbeit liegt der Schwerpunkt auf der Kontextermittlung in Benutzeranwendungen und dem kontext-adaptivem Anwendungsverhalten. Motiviert ist dies durch die Anforderung für die Gestaltung eines kontext-sensitiven Sicherheitsdienstes, die Anwendungsebene als Schnittstelle zu einem Akteur zentral einzubeziehen. Die mobilen Nutzungsszenarien bringen als übergreifendes Szenario die konventionellen Informationssicherheitskonzepte an ihre Grenzen. Kontext-sensitive Sicherheitsdienste stellen durch ihre adäquate Modellierung einem Akteur einen angemesseneren Dienst zu Verfügung. Modellbildend hierbei ist die Sicherheitsrelevanz formuliert durch die Informationssicherheitsziele der zu schützenden Assets.

In der durchgeführten Auseinandersetzung wird eine konzeptuelle Ausgestaltung der Einbeziehung der Anwendungsebene in einen kontext-sensitiven Sicherheitsdienst für mobile Nutzungsszenarien vorgestellt. Der dargelegte Ansatz erlaubt die Interaktion zwischen einem Akteur und einem mobilen Endgerät sicherheitsrelevant und anwendungs-spezifisch zu modellieren, inklusive einer kontext-sensitiven Sicherheitsadaption für die Anwendungsebene.

Die ersten Schritte befassen sich mit der Identifikation der rekurrente Strukturen von Benutzeraktivitäten und deren Sicherheitsrelevanz. In Anlehnung an das Konzept der *Angriffsfläche* [MW11] findet eine Modellierung von Interaktionsmerkmalen statt. Dieses Konzept von Manadhata und Wing [MW11] erlaubt ebenso die kontext-sensitive Sicherheitsadaption konzeptuell zu erfassen. Hierzu wird das dynamische Adaptionsverhalten als Angriffsflächenreduzierung konzeptualisiert vorgestellt. Die weiterführende Modellierung orientiert sich daran und stellt entsprechend im 5. Kapitel, dass dieser Arbeit zugrunde liegen konzeptuelle Modell vor. Darin wird besonders der Korrelation zwischen den Benutzeraktivitäten und der Sicherheitsadaption Aufmerksamkeit

geschenkt.

Um den entwickelten Ansatz zur Einbeziehung von sicherheitsrelevanter Benutzeraktivität zu erden, wird ein infrastrukturelles Modell für die Verarbeitung von Anwendungskontextinformationen im 6. Kapitel vorgestellt. Es wird dazu ein bestehendes Architekturmodell [FK11] zu einer Infrastruktur für kontext-sensitive Sicherheit ausmodelliert. Die aus dieser Konzeption gewonnenen Erkenntnisse werden im darauffolgenden 7. Kapitel für die Modellierung der Interaktions- und Kommunikationsmodelle genutzt. Diese Modelle führen aus, wie die Anwendungsebene in eine Infrastruktur für kontext-sensitive Sicherheit zu integrieren ist. Eine dedizierte Anwendungskomponente (Security Component) kapselt dazu anwendungs-spezifisch die Funktionalität.

Im dritten Abschnitt des 7. Kapitels wird der Ansatz zur Einbeziehung von sicherheitsrelevanter Benutzeraktivität zusammen mit dem kontext-adaptivem Sicherheitsverhalten konkretisiert vorgestellt. Das dynamische Adaptionsverhalten als Angriffsflächenreduzierung wir hierzu ausführlich als *Whitelist*-Ansatz erläutert und die Integration der Anwendungskomponente (Security Component) anhand des MVC Anwendungsentwurfsmusters thematisiert.

Die formale Modellierung des Anwendungskontextes und der Sicherheitsadaption inklusive der Kommunikation selbst, wird entlang des XML-RPC-Protokolls [Win03] in Form eines W3C XML-Schema [W3C12] vorgenommen. Zusätzlich werden anwendungskontext-spezifische Erweiterungen entwickelt und spezifiziert sowie integriert im XML-RPC Datenmodell vorgestellt. In Anbetracht der rudimentären Instanziierung (s. Kap. 7) des vorgestellten Konzepts zur Einbeziehung von sicherheitsrelevanter Benutzeraktivität, soll durch die Wahl des XML-RPC-Protokolls unteranderem eine allgemeine Flexibilität unterstützt werden[1]. Dies adressiert vor allem Anforderungen, die bei konkreten Implementierungen auftreten können. Im vorgestellten Ansatz unterstützt das XML-RPC-Protokoll sowohl eine architektonische als auch eine datenmodellspezifische Flexibilität.

Das vorgestellte Anwendungskontextmodell stellt einen wesentlichen Beitrag zu einer übergreifenden Modellierung eines sicherheitsrelevanten Kontextes bei. Das im Kapitel 5 vorgestellte konzeptuelle Modell hat eine Allge-

[1] nicht-funktionale Anforderung der Erweiterbarkeit

meingültigkeit. Die darauf folgende Auseinandersetzung stellt eine exemplarische Instanziierung des Modells dar. Durch die Integration in eine exemplarische Infrastruktur für kontext-sensitive Sicherheit werden die Eigenschaften des Ansatzes im Zusammenspiel mit den anderen Architekturkomponenten dargelegt. Das kontext-adaptive Sicherheitsverhalten stellt den eigentlichen Sicherheitsdienst für einen Benutzer dar, der von dem vorgestellten Ansatz am *Anfang* mit der Ermittlung sicherheitsrelevanter Benutzeraktivität und am *Ende* des Sicherheitsprozesses mit der kontext-sensitiven Sicherheitsadaption effektiv unterstützt wird.

# A. Anhang

## A.1. Anwendungsintegration

Zur abschliessenden Vervollständigung des Kapitels 7 ab Seite 35, befinden sich in diesem Abschnitt referenzierte Quelltexte.

### A.1.1. Erweitertes XML-RPC Datenmodell

Ein offizielles XML-RPC-*Schema* steht nicht zur Verfügung (s. [Win03]). Das für die Modelle genutzte *Schema* orientiert sich an [MSX16] und ist entsprechend Kapitel 7.3 erweitert worden. Die XML-RPC Beispiele sind strukturell wie folgt definiert [W3C12]:

```
1  <?xml version="1.0" encoding="UTF-8"?>
2
3   <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
4
5
6     <xs:element name="methodCall">
7      <xs:complexType>
8        <xs:all>
9          <xs:element name="methodName">
10           <xs:simpleType>
11             <xs:restriction base="ASCIIString">
12               <xs:pattern value="([A-Za-z0-9]|/|\.|:|_)*" />
13             </xs:restriction>
14           </xs:simpleType>
15         </xs:element>
16         <xs:element minOccurs="0" maxOccurs="1" name="params">
17           <xs:complexType>
18             <xs:sequence>
19              <xs:element minOccurs="2" maxOccurs="2" name="param" type="ParamType" />
20             </xs:sequence>
21           </xs:complexType>
```

```
22          </xs:element>
23        </xs:all>
24      </xs:complexType>
25    </xs:element>
26
27    <xs:element name="methodResponse">
28      <xs:complexType>
29        <xs:choice>
30          <xs:element name="params">
31            <xs:complexType>
32              <xs:sequence>
33                <xs:element name="param" type="ParamType" />
34              </xs:sequence>
35            </xs:complexType>
36          </xs:element>
37          <xs:element name="fault">
38            <xs:complexType>
39              <xs:sequence>
40                <xs:element name="value">
41                  <xs:complexType>
42                    <xs:sequence>
43                      <xs:element name="struct">
44                        <xs:complexType>
45                          <xs:sequence>
46                            <xs:element name="member" type="MemberType" />
47                            <xs:element name="member" type="MemberType" />
48                          </xs:sequence>
49                        </xs:complexType>
50                      </xs:element>
51                    </xs:sequence>
52                  </xs:complexType>
53                </xs:element>
54              </xs:sequence>
55            </xs:complexType>
56          </xs:element>
57        </xs:choice>
58      </xs:complexType>
59    </xs:element>
60
61
62
63    <xs:complexType name="ParamType">
64      <xs:sequence>
65        <xs:element name="value" type="ValueType" />
66      </xs:sequence>
67    </xs:complexType>
68
69    <xs:complexType name="ValueType" mixed="true">
70      <xs:choice>
71        <xs:element name="i4" type="xs:int" />
72        <xs:element name="int" type="xs:int" />
```

```
73        <xs:element name="string" type="ASCIIString" />
74        <xs:element name="double" type="xs:decimal" />
75        <xs:element name="Base64" type="xs:base64Binary" />
76        <xs:element name="boolean" type="NumericBoolean" />
77        <xs:element name="dateTime.iso8601" type="xs:dateTime" />
78        <xs:element name="array" type="ArrayType" />
79        <xs:element name="struct" type="StructType" />
80        <xs:element name="operation" type="ContextActivityOperation" />
81        <xs:element name="seclevelclass" type="ContextActivityAssetSecLevelClass" />
82        <xs:element name="channel" type="ContextActivityChannel" />
83        <xs:element name="mechanism" type="MeasuresMechanism" />
84      </xs:choice>
85    </xs:complexType>
86
87    <xs:complexType name="StructType">
88      <xs:sequence>
89        <xs:element maxOccurs="unbounded" name="member" type="MemberType" />
90      </xs:sequence>
91    </xs:complexType>
92
93    <xs:complexType name="MemberType">
94      <xs:sequence>
95        <xs:element name="name" type="xs:string" />
96        <xs:element name="value" type="ValueType" />
97      </xs:sequence>
98    </xs:complexType>
99
100   <xs:complexType name="ArrayType">
101     <xs:sequence>
102       <xs:element name="data">
103         <xs:complexType>
104           <xs:sequence>
105  <xs:element minOccurs="0" maxOccurs="unbounded" name="value" type="ValueType" />
106           </xs:sequence>
107         </xs:complexType>
108       </xs:element>
109     </xs:sequence>
110   </xs:complexType>
111
112   <xs:complexType name="ContextActivityChannel">
113     <xs:sequence>
114       <xs:element name="class" type="ContextActivityChannelClass" />
115       <xs:element name="type" type="ContextActivityChannelType" />
116     </xs:sequence>
117   </xs:complexType>
118
119
120
121   <xs:simpleType name="ASCIIString">
122     <xs:restriction base="xs:string">
123       <xs:pattern value="([␣-~]|\n|\r|\t)*" />
```

```
124       </xs:restriction>
125     </xs:simpleType>
126
127     <xs:simpleType name="NumericBoolean">
128       <xs:restriction base="xs:boolean">
129         <xs:pattern value="0|1" />
130       </xs:restriction>
131     </xs:simpleType>
132
133     <xs:simpleType name="ContextActivityOperation">
134       <xs:restriction base="ASCIIString">
135     <xs:pattern value="attribute|close|create|delete|move|modify|open|read|write" />
136       </xs:restriction>
137     </xs:simpleType>
138
139     <xs:simpleType name="ContextActivityAssetSecLevelClass">
140       <xs:restriction base="xs:int">
141         <xs:pattern value="0|1|2|3|4|5|6|7|8|9" />
142       </xs:restriction>
143     </xs:simpleType>
144
145     <xs:simpleType name="ContextActivityChannelType">
146       <xs:restriction base="ASCIIString">
147         <xs:pattern value="input|output|inputoutput" />
148       </xs:restriction>
149     </xs:simpleType>
150
151     <xs:simpleType name="ContextActivityChannelClass">
152       <xs:restriction base="ASCIIString">
153         <xs:pattern value="internal|local|externallocal|externalwide|remote" />
154       </xs:restriction>
155     </xs:simpleType>
156
157     <xs:simpleType name="MeasuresMechanism">
158       <xs:restriction base="ASCIIString">
159         <xs:pattern
160          value="authorization|information|alternative|confirmation|authentication" />
161       </xs:restriction>
162     </xs:simpleType>
163
164
165   </xs:schema>
```

Quelltext A.1: Erweitertes XML-RPC-Schema

### A.1.2. Benutzeraktivität

Es folgt entsprechend dem Datenmodell aus Anhang A.1.1 ein validierter Aufruf für die Benachrichtigung des Kontextrahmenswerks (s. Kap. 6.1.2) bezüglich einer sicherheitsrelevanten Benutzeraktivität (XML-RPC):

```
POST /service/appcontext HTTP/1.1
User-Agent: APP_NAME/APP_VERSION
Host: localhost:FRAMEWORK_STATIC_LOWPORT
Content-Type: text/xml
Content-length: 7181

<?xml version="1.0" encoding="UTF-8"?>
<methodCall>
  <methodName>APP_REG_ID.ActivityNotifyCb</methodName>
<!-- START passing values -->
   <params>

<!-- START First argument for metadata of the notify -->
    <param>
      <value>
        <struct>
            <member>
<!-- Application Identifier with own Namespace -->
                <name>ACTIVITY_APPLICATION_ID</name>
                    <value>
                        <string>ORG.SHEE.APPFAMILY.APPNAME.BUILDID</string>
                        </value>
                </member>

            <member>
<!-- Universally Unique Event Identifier -->
                <name>ACTIVITY_UUID</name>
                    <value>
                        <string>c4910502-edeb-e115-c978-c77afd49065f</string>
                        </value>
                </member>
            <member>
<!-- Timestamp (UTC) only valid on local system! -->
                <name>ACTIVITY_APP_TIMESTAMP</name>
                    <value>
                        <dateTime.iso8601>2016-03-18T16:47:29.1Z</dateTime.iso8601>
                        </value>
                </member>

            </struct>
          </value>
        </param>
<!-- END First argument for metadata of the notify -->
```

```
44
45
46  <!-- START 2nd argument for application context -->
47      <param>
48        <value>
49          <struct>
50              <member>
51  <!-- Linear security classification 0-9 -->
52                  <name>ACTIVITY_ASSET</name>
53                    <value>
54                          <seclevelclass>8</seclevelclass>
55                      </value>
56                  </member>
57
58              <member>
59  <!-- Activity operation -->
60                  <name>ACTIVITY_OPERATION</name>
61                      <value>
62                          <operation>read</operation>
63                          </value>
64                  </member>
65
66              <member>
67  <!-- Activity channel -->
68                  <name>ACTIVITY_CHANNEL</name>
69                      <value>
70                        <channel>
71                            <class>externalwide</class>
72                            <type>output</type>
73                        </channel>
74                      </value>
75                  </member>
76
77              <member>
78  <!-- Application capabilities_A (current measures context)-->
79                 <name>CAPABILITIES_ACTIV</name>
80                  <value>
81                    <struct>
82                      <member>
83  <!-- Linear security classification max activ security class is enough -->
84                          <name>ATTRIBUTE_ASSET</name>
85                              <value>
86                                  <array>
87                                      <data>
88                                        <value><seclevelclass>6</seclevelclass></value>
89                                       </data>
90                                    </array>
91                              </value>
92                          </member>
93                      <member>
94  <!-- Example: any write related operation is not listed-->
```

```
95                              <name>ATTRIBUTE_OPERATION</name>
96                                  <value>
97                                     <array>
98                                        <data>
99                                          <value><operation>close</operation></value>
100                                         <value><operation>open</operation></value>
101                                         <value><operation>read</operation></value>
102                                        </data>
103                                     </array>
104                                 </value>
105                             </member>
106                         <member>
107 <!-- No remote channel allowed -->
108                             <name>ATTRIBUTE_CHANNEL</name>
109                                <value>
110                                  <array>
111                                    <data>
112                                      <value>
113                                       <channel>
114                                        <class>internal</class><type>input</type>
115                                       </channel></value>
116                                      <value>
117                                       <channel>
118                                        <class>internal</class><type>output</type>
119                                       </channel>
120                                      </value>
121                                      <value>
122                                       <channel>
123                                        <class>internal</class><type>inputoutput</type>
124                                       </channel>
125                                      </value>
126                                      <value>
127                                       <channel>
128                                        <class>local</class><type>input</type>
129                                       </channel>
130                                      </value>
131                                      <value>
132                                       <channel>
133                                        <class>local</class><type>output</type>
134                                       </channel>
135                                      </value>
136                                      <value>
137                                       <channel>
138                                        <class>local</class><type>inputoutput</type>
139                                       </channel>
140                                      </value>
141                                      <value>
142                                       <channel>
143                                        <class>externallocal</class><type>input</type>
144                                       </channel>
145                                      </value>
```

```
146                                    <value>
147                                     <channel>
148                                      <class>externallocal</class><type>output</type>
149                                     </channel>
150                                    </value>
151                                    <value>
152                                     <channel>
153                                    <class>externallocal</class><type>inputoutput</type>
154                                     </channel>
155                                    </value>
156                                    <value>
157                                     <channel>
158                                      <class>externalwide</class><type>input</type>
159                                     </channel>
160                                    </value>
161                                    <value>
162                                     <channel>
163                                      <class>externalwide</class><type>output</type>
164                                     </channel>
165                                    </value>
166                                    <value>
167                                     <channel>
168                                    <class>externalwide</class><type>inputoutput</type>
169                                     </channel>
170                                    </value>
171                                   </data>
172                                  </array>
173                                 </value>
174                               </member>
175                             </struct>
176                           </value>
177                         </member>
178
179                     </struct>
180                   </value>
181                 </param>
182 <!-- END 2nd argument for application context -->
183
184    </params>
185 <!-- END passing values -->
186 </methodCall>
```

Quelltext A.2: XML-RPC Benutzeraktivitäts Benachrichtigung

### A.1.3. Sicherheitsmassnahmen

Es folgt entsprechend dem Datenmodell aus Anhang A.1.1 ein validierter Aufruf für die Benachrichtigung der Anwendungskomponente (s. Kap. 6.1.1) bezüglich spezifischer Sicherheitsmassnahmen korrespondierend mit der über ACTIVITY_UUID identifizierbaren Benutzeraktivität (XML-RPC):

```
POST /service/appmeasures HTTP/1.1
User-Agent: CONTEXT_FRAMEWORK/FRAMEWORK_VERSION
Host: localhost:APP_RANDOM_HIGHPORT
Content-Type: text/xml
Content-length: 5965

<?xml version="1.0" encoding="UTF-8"?>
<!-- Response Callback for Measures -->
<methodCall>
  <methodName>FRAMEWORK_REG_ID.SecResultNotifyCb</methodName>
<!-- START passing values -->
   <params>

<!-- START First argument for metadata of the security response -->
    <param>
      <value>
        <struct>
            <member>
<!-- Universally Unique Event Reference -->
                <name>MEASURES_REF_ACTIVITY_UUID</name>
                    <value>
                        <string>c4910502-edeb-e115-c978-c77afd49065f</string>
                        </value>
                </member>

            <member>
<!-- Priority of measures => type of ASSET seclevelclass -->
                <name>MEASURES_PRIORITY</name>
                    <value>
                        <seclevelclass>9</seclevelclass>
                        </value>
                </member>
            <member>
<!-- SecClass Achievement  -->
                <name>MEASURES_SECGOAL</name>
                    <value>
                        <boolean>1</boolean>
                        </value>
                </member>

            </struct>
          </value>
```

```
43          </param>
44  <!-- END First argument for metadata of the security response -->
45
46
47  <!-- START 2nd argument for application measures -->
48      <param>
49        <value>
50          <struct>
51            <member>
52  <!-- Allowed Measures Mechanism -->
53              <name>MEASURES_MECHANISM</name>
54               <value>
55                 <array>
56                    <data>
57                      <value><mechanism>alternative</mechanism></value>
58                      <value><mechanism>authorization</mechanism></value>
59                     </data>
60                   </array>
61               </value>
62             </member>
63             <member>
64  <!-- Measures for Application (Capabilities_m) -->
65               <name>CAPABILITIES_MEASURES</name>
66                  <value>
67                    <struct>
68                      <member>
69  <!-- Security classification correspondes with activity class -->
70                          <name>ATTRIBUTE_ASSET</name>
71                              <value>
72                                 <array>
73                                    <data>
74                                      <value><seclevelclass>9</seclevelclass></value>
75                                     </data>
76                                   </array>
77                              </value>
78                          </member>
79                      <member>
80  <!-- Write related operations are not listed-->
81                          <name>ATTRIBUTE_OPERATION</name>
82                              <value>
83                                 <array>
84                                    <data>
85                                      <value><operation>close</operation></value>
86                                      <value><operation>open</operation></value>
87                                      <value><operation>read</operation></value>
88                                     </data>
89                                   </array>
90                              </value>
91                          </member>
92                      <member>
93  <!-- No remote and externalwide channel allowed -->
```

```
94                  <name>ATTRIBUTE_CHANNEL</name>
95                    <value>
96                      <array>
97                        <data>
98                          <value>
99                           <channel>
100                           <class>internal</class><type>input</type>
101                          </channel></value>
102                         <value>
103                          <channel>
104                           <class>internal</class><type>output</type>
105                          </channel>
106                         </value>
107                         <value>
108                          <channel>
109                           <class>internal</class><type>inputoutput</type>
110                          </channel>
111                         </value>
112                         <value>
113                          <channel>
114                           <class>local</class><type>input</type>
115                          </channel>
116                         </value>
117                         <value>
118                          <channel>
119                           <class>local</class><type>output</type>
120                          </channel>
121                         </value>
122                         <value>
123                          <channel>
124                           <class>local</class><type>inputoutput</type>
125                          </channel>
126                         </value>
127                         <value>
128                          <channel>
129                           <class>externallocal</class><type>input</type>
130                          </channel>
131                         </value>
132                         <value>
133                          <channel>
134                           <class>externallocal</class><type>output</type>
135                          </channel>
136                         </value>
137                         <value>
138                          <channel>
139                       <class>externallocal</class><type>inputoutput</type>
140                          </channel>
141                         </value>
142                       </data>
143                     </array>
144                   </value>
```

```
145                         </member>
146                       </struct>
147                     </value>
148                   </member>
149
150               </struct>
151             </value>
152           </param>
153 <!-- END 2nd argument for application measures -->
154
155     </params>
156 <!-- END passing values -->
157 </methodCall>
```

Quelltext A.3: XML-RPC Sicherheitsmassnahmen Benachrichtigung

### A.1.4. Vollständige Anwendungsfähigkeit

Um einen Konsens über mögliche Sicherheitsmassnahmenkonfigurationen herstellen zu können, muss die gesamte anwendungs-spezifische Fähigkeit (*Capabilities*$_{App}$, s. S. 55 ff.) der Infrastruktur für kontext-sensitive Sicherheit zur Verfügung gestellt werden. Bezüglich des Umfangs muss zusätzlich davon ausgegangen werden, dass nicht alle Anwendungen den maximalen Umfang selbst anbieten, der von der vorgestellten Modellierung abgedeckt wird (s. Kap. 7.3.2).

Es folgt entsprechend dem Datenmodell aus Anhang A.1.1 ein validierter Registrierungsaufruf am Kontextrahmenwerk (s. Kap. 6.1.1), das dabei die anwendungs-spezifischen Fähigkeiten kommuniziert bekommt. Um einen verminderten Umfang darzulegen, zeigt das Beispiel eine Anwendung, die keine Unterscheidung zwischen Dateneingang und Ausgang bei *externen* und *entfernten* Kommunikationskanälen vornimmt (XML-RPC):

```
POST /service/appcontext HTTP/1.0
User-Agent: APP_NAME/APP_VERSION
Host: localhost:FRAMEWORK_STATIC_LOWPORT
Content-Type: text/xml
Content-length: 4274

<?xml version="1.0" encoding="UTF-8"?>
<methodCall>
  <methodName>PublicServiceChannelforMeasures</methodName>
<!-- START passing values -->
   <params>

<!-- START First argument for metadata of the registration -->
    <param>
      <value>
        <struct>
            <member>
<!-- Application Identifier with own Namespace -->
                <name>MEASURES_APPLICATION_ID</name>
                    <value>
                     <string>ORG.SHEE.APPFAMILY.APPNAME.BUILDID</string>
                     </value>
                </member>
            <member>
<!-- Applications Callback for measures -->
                <name>MEASURES_CALLBACK</name>
                    <value>
         <string>http://localhost:APP_RANDOM_HIGHPORT/service/appmeasures</string>
                     </value>
                </member>
```

```
31
32              </struct>
33            </value>
34          </param>
35 <!-- END First argument for metadata of the notify -->
36
37
38 <!-- START 2nd argument for application capabilities informations -->
39         <param>
40          <value>
41           <struct>
42             <member>
43              <!-- Application capabilities_App -->
44              <name>CAPABILITIES_APP</name>
45              <value>
46                <struct>
47                 <member>
48 <!-- Linear class; maximum asset security class enough (scope 0-max) -->
49                  <name>ATTRIBUTE_ASSET</name>
50                  <value>
51                   <array>
52                    <data>
53                      <value><seclevelclass>9</seclevelclass></value>
54                     </data>
55                   </array>
56                  </value>
57                 </member>
58                 <member>
59 <!-- All operations -->
60                  <name>ATTRIBUTE_OPERATION</name>
61                  <value>
62                   <array>
63                    <data>
64                      <value><operation>attribute</operation></value>
65                      <value><operation>close</operation></value>
66                      <value><operation>create</operation></value>
67                      <value><operation>delete</operation></value>
68                      <value><operation>move</operation></value>
69                      <value><operation>modify</operation></value>
70                      <value><operation>open</operation></value>
71                      <value><operation>read</operation></value>
72                      <value><operation>write</operation></value>
73                     </data>
74                   </array>
75                  </value>
76                 </member>
77                 <member>
78 <!-- Not capable in distinguishing in/out for external and remote channels -->
79                  <name>ATTRIBUTE_CHANNEL</name>
80                  <value>
81                   <array>
```

```
82                    <data>
83                     <value>
84                      <channel><class>internal</class><type>input</type></channel>
85                     </value>
86                     <value>
87                      <channel><class>internal</class><type>output</type></channel>
88                     </value>
89                     <value>
90                   <channel><class>internal</class><type>inputoutput</type></channel>
91                     </value>
92                     <value>
93                      <channel><class>local</class><type>input</type></channel>
94                     </value>
95                     <value>
96                      <channel><class>local</class><type>output</type></channel>
97                     </value>
98                     <value>
99                     <channel><class>local</class><type>inputoutput</type></channel>
100                    </value>
101                    <value>
102            <channel><class>externallocal</class><type>inputoutput</type></channel>
103                    </value>
104                    <value>
105             <channel><class>externalwide</class><type>inputoutput</type></channel>
106                    </value>
107                    <value>
108                   <channel><class>remote</class><type>inputoutput</type></channel>
109                    </value>
110                   </data>
111                  </array>
112                 </value>
113                </member>
114               </struct>
115              </value>
116             </member>
117           </struct>
118          </value>
119         </param>
120 <!-- END 2nd argument for application capabilities informations -->
121
122    </params>
123 <!-- END passing values -->
124 </methodCall>
```

Quelltext A.4: XML-RPC Anwendungsfähigkeiten Benachrichtigung

# Literaturverzeichnis


[ABKS09] AN, GAEIL, GUNTAE BAE, KIYOUNG KIM und DONGIL SEO: *Context-aware Dynamic Security Configuration for Mobile Communication Device*. In: *Proceedings of the 3rd International Conference on New Technologies, Mobility and Security*, NTMS'09, Seiten 79–83, Piscataway, NJ, USA, 2009. IEEE Press.

[ADB+99] ABOWD, GREGORY D., ANIND K. DEY, PETER J. BROWN, NIGEL DAVIES, MARK SMITH und PETE STEGGLES: *Towards a Better Understanding of Context and Context-Awareness*. In: *Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing*, HUC '99, Seiten 304–307, London, UK, 1999. Springer-Verlag.

[Ay07] AY, FERUZAN: *Context Modeling and Reasoning using Ontologies*. University of Technology Berlin, July 2007.

[BB05] BAZIRE, MARY und PATRICK BRÉZILLON: *Understanding Context Before Using It*. In: DEY, ANIND, BOICHO KOKINOV, DAVID LEAKE und ROY TURNER (Herausgeber): *Modeling and Using Context*, Band 3554 der Reihe *Lecture Notes in Computer Science*, Seiten 29–40. Springer Berlin Heidelberg, 2005.

[BBH+09] BETTINI, CLAUDIO, OLIVER BRDICZKA, KAREN HENRICKSEN, JADWIGA INDULSKA, DANIELA NICKLAS, ANAND RANGANATHAN und DANIELE RIBONI: *A Survey of Context Modelling and Reasoning Techniques*. Pervasive Mob. Comput., 6(2):161–180, April 2009.

[BDR07] BALDAUF, MATTHIAS, SCHAHRAM DUSTDAR und FLORIAN ROSENBERG: *A survey on context-aware systems*. In: *Int. J. Ad Hoc and Ubiquitous Computing*, Band 2, Seiten 263–277, 2007.

[BGF+10] BAI, GUANGDONG, LIANG GU, TAO FENG, YAO GUO und XIANGQUN CHEN: *Context-Aware Usage Control for Android*. In: JAJODIA, SUSHIL und JIANYING ZHOU (Herausgeber): *Security and Privacy in Communication Networks*, Band 50 der Reihe *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, Seiten 326–343. Springer Berlin Heidelberg, 2010.

[BMR+96] BUSCHMANN, FRANK, REGINE MEUNIER, HANS ROHNERT, PETER SOMMERLAD und MICHAEL STAL: *Pattern-Oriented Software Architecture: A System of Patterns*, Band 1. Ashish Raut, 1996.

[BPSM+06] BRAY, TIM, JEAN PAOLI, C. M. SPERBERG-MCQUEEN, EVE MALER, FRANÇOIS YERGEAU und JOHN COWAN (ED.): *Extensible Markup Language (XML) 1.1 (Second Edition), W3C Recommendation*. https://www.w3.org/TR/2006/REC-xml11-20060816/, August 2006.

[BPVG09] BANDINELLI, MATTEO, FEDERICA PAGANELLI, GIANLUCA VANNUCCINI und DINO GIULI: *A Context-Aware Security Framework for Next Generation Mobile Networks*. In: SCHMIDT, ANDREASU. und SHIGUO LIAN (Herausgeber): *Security and Privacy in Mobile Information and Communication Systems*, Band 17 der Reihe *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, Seiten 134–147. Springer Berlin Heidelberg, 2009.

[CCB08] CUPPENS, FRÉDÉRIC und NORA CUPPENS-BOULAHIA: *Modeling contextual security policies*. International Journal of Information Security, 7(4):285–305, 2008.

[CDK03] COULOURIS, G.F., J. DOLLIMORE und T. KINDBERG: *Verteilte Systeme: Konzepte und Design*. Informatik - Pearson Studium. Pearson Education Deutschland, 2003.

[CFZA02] COVINGTON, MICHAEL J., PRAHLAD FOGLA, ZHIYUAN ZHAN und MUSTAQUE AHAMAD: *A Context-Aware Security Architecture for Emerging Applications*. In: *Proceedings of the 18th Annual Computer Security Applications Conference*, ACSAC '02, Seiten 249–, Washington, DC, USA, 2002. IEEE Computer Society.

[CK00] CHEN, GUANLING und DAVID KOTZ: *A Survey of Context-Aware Mobile Computing Research*. Technischer Bericht TR2000-381, Dartmouth College, Computer Science, Hanover, NH, November 2000.

[CLM14] CABRI, GIACOMO, MAURO LEONCINI und RICCARDO MARTOGLIA: *AMBIT: Towards an Architecture for the Development of Context-dependent Applications and Systems*. In: *Proceedings of the 3rd International Conference on Context-Aware Systems and Applications*, ICCASA '14, Seiten 64–68, ICST, Brussels, Belgium, Belgium, 2014. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[CMT07] CORRADI, ANTONIO, REBECCA MONTANARI und ALESSANDRA TONINELLI: *Adaptive Semantic Middleware for Mobile Environments*. Journal of Networks, 2(1), 02 2007.

[DAS01] DEY, ANIND K., GREGORY D. ABOWD und DANIEL SALBER: *A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-aware Applications*. Hum.-Comput. Interact., 16(2):97–166, Dezember 2001.

[DIN98] *DIN EN ISO 9241-11: Ergonomische Anforderungen für Bürotätigkeiten mit Bildschirmgeräten*, Januar 1998.

[DIN06] *DIN EN ISO 9241-110: Ergonomie der Mensch-System-Interaktion - Grundsätze der Dialoggestaltung*, September 2006.

[DIN08] *DIN ISO IEC 27001: Informationstechnik – IT-Sicherheitsverfahren – Informationssicherheits-Managementsysteme – Anforderungen*, September 2008.

[DIN11] *DIN ISO IEC 27000: Informationstechnik – IT-Sicherheitsverfahren – Informationssicherheits-Managementsysteme – Überblick und Terminologie*, Juli 2011.

[EPS10] EVESTI, ANTTI und SUSANNA PANTSAR-SYVÄNIEMI: *Towards Micro Architecture for Security Adaptation*. In: *Proceedings of the Fourth European Conference on Software Architecture: Companion Volume*, ECSA '10, Seiten 181–188, New York, NY, USA, 2010. ACM.

[FGM+99] FIELDING, R., J. GETTYS, J. MOGUL, H. FRYSTYK, L. MASINTER, P. LEACH und TIM B. LEE: *RFC 2616 - HTTP/1.1, the Hypertext Transfer Protocol*. https://www.ietf.org/rfc/rfc2616.txt, 1999.

[FK11] FISCHER, KRISTIAN und STEFAN KARSCH: *Modelling Security Relevant Context An approach towards Adaptive Security in Volatile Mobile Web Environments*. In: *Proceedings of the ACM WebSci'11*, Band WebSci Conference 2011, Seiten 1–3, Juni 2011.

[FKRL09] FRANK, KORBINIAN, NIKOS KALATZIS, IOANNA ROUSSAKI und NICOLAS LIAMPOTIS: *Challenges for Context Management Systems Imposed by Context Inference*. In: *Proceedings of the 6th International Workshop on Managing Ubiquitous Communications and Services*, MUCS '09, Seiten 27–34, New York, NY, USA, 2009. ACM.

[FMS01] FLUHRER, SCOTT, ITSIK MANTIN und ADI SHAMIR: *Selected Areas in Cryptography: 8th Annual International Workshop, SAC 2001 Toronto, Ontario, Canada, August 16–17, 2001 Revised Papers*, Kapitel Weaknesses in the Key Scheduling Algorithm of RC4, Seiten 1–24. Springer Berlin Heidelberg, Berlin, Heidelberg, 2001.

[GHJV94] GAMMA, E., R. HELM, R. JOHNSON und J. VLISSIDES: *Design Patterns: Elements of Reusable Object-Oriented Software*. Pearson Education, 1994.

[HM06] HÄKKILÄ, JONNA und JANI MÄNTYJÄRVI: *Developing Design Guidelines for Context-aware Mobile Applications*. In: *Proceedings of the 3rd International Conference on Mobile Technology, Applications & Systems*, Mobility '06, New York, NY, USA, 2006. ACM.

[IAN16] *IANA: Service Name and Transport Protocol Port Number Registry*. https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.txt, 2016.

[iOS15] *iOS Developer Library, Cocoa Core Competencies: Model-View-Controller*. https://developer.apple.com/library/ios/documentation/General/Conceptual/DevPedia-CocoaCore/MVC.html, 2015.

[JAB10] JOHNSON, GLENEESHA, ASHOK AGRAWALA und ELODIE BILLIONNIERE: *A Framework for Shrink-Wrapping Security Services*. In: *Proceedings of the 2010 IEEE International Conference on Services Computing*, SCC '10, Seiten 639–640, Washington, DC, USA, 2010. IEEE Computer Society.

[JGK14] JOVANOVIKJ, VLADIMIR, DUŠAN GABRIJELČIČ und TOMAŽ KLOBUČAR: *A Conceptual Model of Security Context*. Int. J. Inf. Secur., 13(6):571–581, November 2014.

[Joh09] JOHNSON, GLENEESHA M.: *Towards Shrink-wrapped Security: A Taxonomy of Security-relevant Context*. In: *Proceedings of the 2009 IEEE International Conference on Pervasive Computing and Communications*, PERCOM '09, Seiten 1–2, Washington, DC, USA, 2009. IEEE Computer Society.

[Kec06] KECHER, CHRISTOPH: *UML 2.0: Das umfassende Handbuch*. Galileo computing. Galileo Press, 2006.

[Kie12] KIEFER, PETER: *The Mobile Intention Recognition Problem And An Approach Based On Spatially-Constrained Grammars*. Springer New York, 2012.

[Kob12] KOBINSKI, RAFAEL: *Eine Architektur zur Ableitung und Umsetzung kontextabhängiger Sicherheitsmaßnahmen auf mobilen Geräten*. Diplomarbeit, Fachhochschule Köln, 2012.

[KP88] KRASNER, GLENN E. und STEPHEN T. POPE: *A Cookbook for Using the Model-view Controller User Interface Paradigm in Smalltalk-80*. J. Object Oriented Program., 1(3):26–49, August 1988.

[Mar83] MARTIN, J.: *Managing the Data Base Environment*. A James Martin book. Pearson Education, Limited, 1983.

[MATA+14] MOWAFI, YASER, DHIAH ABOU-TAIR, TAREQ AQARBEH, MARAT ABILOV, VIKTOR DMITRIYEV und JORGE MARX GOMEZ: *A Context-aware Adaptive Security Framework for Mobile Applications*. In: *Proceedings of the 3rd International Conference on Context-Aware Systems and Applications*, ICCASA '14, Seiten 147–153, ICST, Brussels, Belgium, Belgium, 2014. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[MB03] MOSTÉFAOUI, GHITA KOUADRI und PATRICK BRÉZILLON: *Modeling and Using Context: 4th International and Interdisciplinary Conference CONTEXT 2003 Stanford, CA, USA, June 23–25, 2003 Proceedings*, Kapitel A Generic Framework for Context-Based Distributed Authorizations, Seiten 204–217. Springer Berlin Heidelberg, Berlin, Heidelberg, 2003.

[MB04] MOSTEFAOUI, G.K. und P. BREZILLON: *Modeling context-based security policies with contextual graphs*. In: *Pervasive Computing and Communications Workshops, 2004. Proceedings of the Second IEEE Annual Conference on*, Seiten 28–32, March 2004.

[MSX16] *Microsoft Technical Documents: XML-RPC Schema*. https://msdn.microsoft.com/en-us/library/ff797558(v=office.12).aspx, 2016.

[Mül10] MÜLLER, FELIX: *Modellierung und Repräsentation des Kontextes von mobilen Nutzungsszenarien - ein Rahmenwerk für mobile kontextsensitive Applikationen*. Masterthesis, Fachhochschule Köln, 2010.

[MW11] MANADHATA, P.K. und J.M. WING: *An Attack Surface Metric*. Software Engineering, IEEE Transactions on, 37(3):371–386, Mai 2011.

[Nor99] NORMAN, DONALD A.: *Affordance, Conventions, and Design*. interactions, 6(3):38–43, Mai 1999.

[Ree79] REENSKAUG, TRYGVE: *Model-View-Controller (MVC)*. https://-heim.ifi.uio.no/~trygver/themes/mvc/mvc-index.html, 1979.

[RFC81] *RFC 793 - Transmission Control Protocol (TCP)*. https://rfc-editor.org/rfc/rfc793.txt, 1981.

[RQ12] RUPP, C. und S. QUEINS: *UML 2 glasklar: Praxiswissen für die UML-Modellierung*. Hanser, 4., aktualisierte und erweiterte Auflage Auflage, 2012.

[SP03] SANDHU, RAVI und JAEHONG PARK: *Usage control: A vision for next generation access control*. In: *MMM-ACNS*, Seiten 17–31, 2003.

[SRC84] SALTZER, J. H., D. P. REED und D. D. CLARK: *End-to-end Arguments in System Design*. ACM Trans. Comput. Syst., 2(4):277–288, November 1984.

[ST94] SCHILIT, B. N. und M. M. THEIMER: *Disseminating active map information to mobile hosts*. IEEE Network, 8(5):22–32, Sept 1994.

[VM10] VILLEGAS, NORHA M. und HAUSI A. MÜLLER: *Managing Dynamic Context to Optimize Smart Interactions and Services*. In: CHIGNELL, MARK, JAMES CORDY, JOANNA NG und YELENA YESHA (Herausgeber): *The Smart Internet: Current Research and Future Applications*, Kapitel Managing Dynamic Context to Optimize Smart Interactions and Services, Seiten 289–318. Springer-Verlag, Berlin, Heidelberg, 2010.

[VV13] VACCA, JOHN R. und JOHN R. VACCA: *Computer and Information Security Handbook, Second Edition*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2nd Auflage, 2013.

[W3C12] W3C: *XML Schema Definition Language (XSD) 1.1 Part 1: Structures*. https://www.w3.org/TR/xmlschema11-1/, April 2012.

[Win03] WINER, DAVE: *Extensible Markup Language Remote Procedure Call (XML-RPC) Specification*. http://xmlrpc.scripting.com/spec.html, 2003.

[YDM12] YE, JUAN, SIMON DOBSON und SUSAN MCKEEVER: *Situation identification techniques in pervasive computing: A review*. Pervasive and Mobile Computing, 8(1):36 – 66, 2012.

# Index

# Erklärungen

## Eidesstattliche Erklärung

Ich versichere, die von mir hier vorgelegte Masterthesis selbständig verfasst zu haben. Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder nicht veröffentlichten Arbeiten anderer entnommen sind, habe ich als entnommen kenntlich gemacht. Sämtliche Quellen und Hilfsmittel, die ich für die Arbeit benutzt habe, sind angegeben. Diese Masterthesis hat mit gleichem Inhalt bzw. in wesentlichen Teilen noch keiner anderen Prüfungsbehörde vorgelegen.

Köln, den 4. Mai 2016

(Vitor A. Guerreiro)

## Sperrvermerk

Die vorgelegte Masterthesis unterliegt keinem Sperrvermerk.

## Weitergabeerklärung

Ich erkläre hiermit mein Einverständnis, dass das vorliegende Exemplar dieser Masterthesis oder eine Kopie hiervon für wissenschaftliche Zwecke verwendet werden darf.

Köln, den 4. Mai 2016

(Vitor A. Guerreiro)